P. 1877/87

4 1987

informatyka

Ocena działania systemów komputerowych

Składnia języka Logo

Mikroprogramowanie

Funkcje systemu BIOS dla PC/XT

Porównanie systemów BIOS dla IBM PC

WYDAWNICTWO NOT SIGMA

Cena 150 zł

informatyka

4 1987

Nr 4
Miesięcznik Rok XXI
Kwiecień 1987

Organ Komitetu
Naukowo-Technicznego NOT
ds. Informatyki

KOLEGIUM REDAKCYJNE:

Dr inż. Wacław ISZKOWSKI, mgr Teresa JABŁOŃSKA (sekretarz redakcji), Władysław KLEPACZ (redaktor naczelny), dr inż. Marek MACHURA, Maria PAWLAK (sekretarz redakcji), mgr inż. Jan RYŻKO, dr inż. Wiktor RZECZKOWSKI, mgr Hanna WŁODARSKA, dr inż. Janusz ZALEWSKI (zastępca red. naczelnego)

PRZEWODNICZĄCY
RADY PROGRAMOWEJ:

Prof. dr hab. Juliusz Lech KULIKOWSKI

Materiałów nie zamówionych redakcja nie zwraca

Redakcja: 01-517 Warszawa, ul. Mickiewicza 18 m. 17, tel. 39-14-34

Zakł. Graf. „Tamka". Zam. 1210-1300/86. Obj. 4,0 ark. druk. Nakład 8050 egz. K-87.

ISSN 0542-9951. INDEKS 36124

Cena egzemplarza 150 zł
Prenumerata roczna 1800 zł

WYDAWNICTWO NACZELNA ORGANIZACJA TECHNICZNA
SIGMA
CZASOPISM I KSIĄŻEK TECHNICZNYCH
00-950 Warszawa
skrytka pocztowa 1004
ul. Biała 4

W NUMERZE: Strona

Ocena działania systemów komputerowych — przedmiot, metody, kierunki rozwoju (1)
Jan Węglarz 1

Składnia języka Logo — próba formalizacji (1)
Ewa Gurbiel, Helena Krupicka, Zdzisław Płoski 5

Programowanie logiczne i komputery nowej generacji (2)
Robert Kowalski 8

Mikroprogramowanie (1)
Marek Pawłowski 11

Funkcje systemu BIOS dla PC/XT
Andrzej Rażnowiecki, Anna Syfert 15

Porównanie systemów BIOS dla rodziny komputerów IBM PC
Janusz Zalewski 18

Szczególnie istotne adresy, przerwania i porty IBM PC/XT
Waldemar Dworakowski 21

System relacyjnej bazy danych SELKO dla minikomputera MERA 400
Maria Meler-Kapcia 23

SAMOTESTY 24
V. A-C. Bazy danych

ZE ŚWIATA 26
Janus/Ada — podzbiór Ady dla mikrokomputerów
Kostki z arsenku galu
Kto jest kim w IFIP — Blagovest H. Sendov
Stan usług w dziedzinie banków danych w Japonii
Tłumaczenie maszynowe

TERMINOLOGIA 31
Polskie nazewnictwo znaków kodu ASCII/ISO (2)

W NAJBLIŻSZYCH NUMERACH:

- Jan Madey omawia problematykę systemów operacyjnych, na przykładzie systemu Unix
- Zbigniew Szkaradnik i Wojciech Petrykowski o grafice żółwia w języku Forth
- Czesław Ratka o adaptacji drukarki wierszowej DW 204-2 do pracy z mikrokomputerem SM-4
- Mariusz Kuc o kompilatorze Ady dla mikrokomputerów
- Marek Machura o systemach ekspertowych na mikrokomputery IBM PC
- Jan Bielecki o definiowaniu funkcji ze zmienną liczbą argumentów w języku C

JAN WĘGLARZ
Instytut Automatyki
Politechnika Poznańska

POLSKIE TOWARZYSTWO INFORMATYCZNE

Pierwsza wersja artykułu została przedstawiona podczas Szkoły Jesiennej PTI „Współczesne kierunki rozwoju informatyki" — Rydzyna, 9—13 października 1984 r.

P. 1877/87

Ocena działania systemów komputerowych — przedmiot, metody, kierunki rozwoju (I)

Z punktu widzenia teorii systemów każdy system jest, mówiąc ogólnie, całością złożoną z wyodrębnionych części (podsystemów), powiązanych w określony sposób do realizacji przyjętych celów w określonym otoczeniu. Zagwarantowanie, przynajmniej potencjalne, realizowania przez system jego celów, stanowi zatem naczelne zadanie jego projektanta, stopień oraz potencjalne możliwości realizacji celów — podstawę do decyzji nabywcy, a maksymalne wykorzystanie potencjalnych możliwości w danych warunkach — główną troskę właściciela. W każdej z powyższych sytuacji, czy ogólniej — na każdym z powyższych etapów, zachodzi konieczność oceny systemu (lub jego modelu) z punktu widzenia efektywności działania. Pytanie o efektywność musi zostać poprzedzone pytaniem o warunki konieczne, a więc o poprawność działania systemu, np. o stabilność systemu automatycznej regulacji, czy o równowagę statystyczną systemu komputerowego jako systemu masowej obsługi.

Ocena działania[1]) systemu ma zatem podstawowe znaczenie w procesie jego projektowania, eksploatacji i rozwoju. Choć wniosek ten istotnie jest bezsporny, to jednak jego realizacja wygląda różnie dla różnych rodzajów systemów. Fakt ten jest obiektywnie uzasadniony różną możliwością budowy modeli matematycznych różnych systemów.

Dlatego też bardzo zaawansowany jest rozwój **teorii i praktyki badań ewaluacyjnych** w dziedzinie systemów mechanicznych, elektrodynamicznych, termodynamicznych czy ogólniej — systemów opisywanych określonymi typami równań operatorowych. Wszak niczym innym, jak działem badań ewaluacyjnych jest, na przykład, projektowanie systemów automatycznej regulacji (liniowych, nieliniowych, ciągłych, dyskretnych, stacjonarnych, niestacjonarnych, o parametrach skupionych i o parametrach rozłożonych) czy jego bardziej zaawansowane stadium — teoria sterowania optymalnego.

W wypadku systemów komputerowych sytuacja wyjściowa do badań ewaluacyjnych jest oczywiście znacznie gorsza. Mamy w nich bowiem do czynienia z wieloma całkowicie różnymi aspektami, przy czym tylko niektóre (i to przy licznych założeniach) dadzą się opisać ilościowo.

[1]) Pomijamy uzasadnienie powodów (dyskusyjnych), dla których termin angielski „performance evaluation" nie tłumaczymy jako „ocena wydajności" lub „ocena osiągów".

Prof. dr hab. inż. JAN WĘGLARZ ukończył studia w 1969 r. na Wydziale Matematyki, Fizyki i Chemii Uniwersytetu im. Adama Mickiewicza w Poznaniu, a w 1971 r. na Wydziale Elektrycznym Politechniki Poznańskiej. W tej samej uczelni uzyskał doktorat (1974) i habilitację (1977). Obecnie jest kierownikiem Zakładu Badań Operacyjnych i Systemów Komputerowych w Instytucie Automatyki. Naukowo zajmuje się badaniami operacyjnymi (rozdział zasobów, szeregowanie zadań) i ich zastosowaniami. Jest autorem wielu publikacji krajowych i zagranicznych.

PRZEDMIOT BADAŃ EWALUACYJNYCH

Przedmiotem badań ewaluacyjnych systemów komputerowych są metody uzyskiwania informacji o działaniu i metody wpływania na działanie tych systemów w celu zapewnienia ich poprawnej i efektywnej — w określonym sensie — pracy w danych warunkach. Warunki te są określone przede wszystkim przez obciążenie systemu, a więc to wszystko, co pojawia się z otoczenia systemu na wejściu. Już z tego ogólnego określenia przedmiotu badań ewaluacyjnych wynika, że istotną w nich rolę odgrywają kryteria oceny działania, jak również modelowania obciążenia systemu. Zagadnieniom tym poświęcimy dwa odrębne punkty.

W pierwszym punkcie przedstawimy klasyfikację problemów ewaluacyjnych i wskażemy obiekty i ich charakterystyki, będące głównymi przedmiotami badań ewaluacyjnych. Najpierw należy jednak poczynić kilka niezbędnych uściśleń zakresu rozważań. Otóż po pierwsze, rozpatrzymy jedynie systemy scentralizowane, z wyjątkiem ostatniego punktu, w którym wspomniano o systemach rozproszonych. Oznacza to, że nie podejmiemy problemów ewaluacyjnych specyficznych dla tych ostatnich systemów, choć praktycznie wszystkie pozostałe rozważania odnoszą się również do nich. Po drugie, nie poruszymy problematyki oceny działania samych programów, których efektywność, mierzona choćby czasem wykonywania i zapotrzebowaniem na pamięć, ma oczywiście wpływ na ocenę działania systemu jako całości. Dekompozycja wyodrębniająca optymalizację samych programów jest jednak dość naturalna, zwłaszcza w systemach ogólnego przeznaczenia.

Należy zwrócić w tym miejscu uwagę na oczywisty fakt, że rozwiązanie, a nawet ścisłe sformułowanie ogólnego problemu zapewnienia najwłaściwszego funkcjonowania systemu komputerowego jako całości w określonych warunkach jest praktycznie niemożliwe. Byłby to bowiem niezwykle złożony problem znalezienia najwłaściwszego (jak go zdefiniować?) kompromisu między wieloma przeciwstawnymi sobie kryteriami, przy niezdeterminowanym opisie przynajmniej niektórych fragmentów systemu i powiązań między nimi. W większości wypadków tak ogólne przedstawienie problemu nie jest na szczęście potrzebne — sytuacja praktyczna narzuca zwykle jedno główne kryterium oceny (inne, wśród nich zawsze koszt, są uwzględniane w ograniczeniach) i jeden lub kilka fragmentów czy zasobów systemu, mających na nie największy wpływ.

Nie oznacza to negowania potrzeby podejścia globalnego, uwzględniającego w rozsądnych i uzasadnionych granicach różne aspekty działania systemu, tym niemniej rozsądna dekompozycja pozostanie zawsze cechą badań ewaluacyjnych. Inna sprawa, że dekompozycja ta bywa często wykonywana nieumiejętnie, bez dostatecznego uwzględnienia wpływu innych podsystemów z tego samego lub z innych poziomów hierarchii czy innych aspektów funkcjonowania systemu (np. nieuwzględnianie narzutu czasowego systemu operacyjnego przy konstruowaniu algorytmów zarządzania zasobami).

Istotą punktu widzenia badań ewaluacyjnych jest pytanie, jak dobrze system lub jego fragment spełnia swoje funkcje. Oczywiście pytanie to trzeba stawiać już na etapie projektowania systemu. Niestety, fakt ten często nie

dociera do projektantów systemów komputerowych, którzy liczą na to, że żądany poziom działania systemu zawsze będzie można uzyskać przez jego późniejsze modyfikacje. Jednakże, przy złym zaprojektowaniu systemu pierwotnego musi to prowadzić do modyfikacji sprzętowych, co nawet pomijając koszt, implikuje zwykle istotne zmiany w oprogramowaniu. Trzeba ponadto podkreślić, że na ogół na drodze wyłącznie sprzętowej nie da się uzyskać żądanych wartości kryteriów oceny działania systemu. Dobrze zaprojektowany system można natomiast często dostroić do obciążenia, nie uciekając się wcale do zmian sprzętowych. Osiągi systemu i jego podatność na modyfikacje stanowią (oczywiście oprócz ceny) główne kryteria, którymi kieruje się nabywca, wybierając spośród systemów dostępnych na rynku.

Można zatem wyróżnić następujące trzy klasy problemów ewaluacyjnych:

- projektowanie,
- wybór,
- ulepszanie.

Kolejność, w jakiej wymieniono te problemy, ma ukazywać logiczne następstwo ich rozwiązywania, choć rozwiązują je różne osoby. Problemy ulepszania są rozumiane szeroko i obejmują, obok strojenia, wszelkie zmiany sprzętowe. Zaliczenie problemu do jednej z powyższych klas zależy od kontekstu. Przykładowo, problem określenia rozmiaru pamięci operacyjnej może być rozpatrywany w trzech aspektach:

- jako problem projektowania, gdy należy określić minimalny rozmiar pamięci zapewniający określony poziom działania systemu dla danego zastosowania,
- jako problem wyboru, gdy porównuje się rozmiar pamięci oferowanych systemów z punktu widzenia potrzeb nabywcy,
- jako problem ulepszania, gdy rozmiar pamięci jest czynnikiem ograniczającym efektywność działania systemu dla danego obciążenia.

Wynika stąd podobieństwo, a nawet zbieżność metod stosowanych do rozwiązywania poszczególnych problemów.

Podana klasyfikacja, zaproponowana przez Ferrariego [5], nie jest oczywiście jedyną klasyfikacją problemów ewaluacyjnych. Reprezentowany w niej punkt widzenia jest jednak na tyle istotny, że jest ona powszechnie stosowana. Warto wspomnieć jeszcze o drugim, uniwersalnym punkcie widzenia, wyróżniającym problemy analizy i syntezy. W wypadku systemów komputerowych mało jest metod syntezy — projektowania systemu spełniającego określone wymagania efektywnościowe wykonuje się na ogół na drodze rozwiązywania problemów analizy modeli tego systemu dla różnych wartości parametrów tych modeli.

Tabela 1. Przykładowe obiekty i ich charakterystyki, będące przedmiotem badań ewaluacyjnych

Obiekty	Charakterystyki
Parametry systemu operacyjnego	Stopień wieloprogramowości, rozmiary buforów, kwant czasu, rozmiar strony
Algorytmy zarządzania zasobami	Algorytmy szeregowania, algorytmy zarządzania pamięcią, metody dostępu do pamięci zewnętrznych, algorytmy zarządzania urządzeniami zewnętrznymi
Połączenia	Kanały i urządzenia zewnętrzne
Rozmieszczenie informacji	Rozmieszczenie zbiorów na różnych poziomach pamięci, rezydentne moduły systemu operacyjnego
Polityka cen	Opłaty za programy nieefektywne, opłaty za wykorzystanie najbardziej obciążonych zasobów, zachęty do wykorzystywania okresów małego obciążenia
Polityka przyjmowania zadań	Wykluczanie niektórych rodzajów zadań lub użytkowników, ograniczanie obciążenia

W tabeli 1 podano przykładowe obiekty i ich charakterystyki, będące najczęściej przedmiotem badań ewaluacyjnych, zarówno w dziedzinie problemów projektowania, jak i ulepszania. Pominięto problemy wyboru, jako mniej podatne na analizę ilościową; nie wymieniono również poszczególnych składników sprzętu, z których każdy może być oczywiście przedmiotem badań ewaluacyjnych z punktu widzenia liczby jednostek i parametrów techniczno-eksploatacyjnych.

Z oceną działania systemów komputerowych, jak zresztą każdej innej klasy systemów, wiąże się ich **modelowanie.** Jest to związek tak ścisły, że słowo „modelowanie" często pomija się w nazwie tej dyscypliny informatyki, a czasem przeciwnie — stanowi ono hasło obejmujące badania ewaluacyjne. Trzeba bowiem pamiętać, że nawet w wypadku metod pomiarowych, gdy informacja pochodzi bezpośrednio z systemu, znajduje się on na ogół pod obciążeniem będącym modelem obciążenia rzeczywistego.

KRYTERIA OCENY DZIAŁANIA SYSTEMÓW KOMPUTEROWYCH

Kryteria (wskaźniki), według których ocenia się jakość działania systemów komputerowych, mogą być bardzo różnorodne. Znaczną ich liczbę stanowią kryteria trudne lub niemożliwe do ilościowego wymierzenia, np. „siła" zbioru (listy) rozkazów, łatwość obsługi, dostępność serwisu, możliwość rozwoju itd. Takie kryteria są trudne do ścisłego wymierzenia, choć oczywiście są brane pod uwagę w problemach ewaluacyjnych, zwłaszcza w wielokryterialnych problemach wyboru. Zasadniczo jednak w badaniach ewaluacyjnych rozważa się kryteria wymierne. Można je podzielić [13] na **zewnętrzne** i **wewnętrzne,** zwane też odpowiednio kryteriami skuteczności (ang. effectiveness) i sprawności (ang. efficiency) systemu. Kryteria zewnętrzne są widziane przez użytkownika systemu i, ogólnie mówiąc, charakteryzują zdolność przetworzenia danego obciążenia i zdolność do spełnienia wymagań czasowych użytkownika. Natomiast kryteria wewnętrzne charakteryzują obiektywną sprawność systemu, pomagając zidentyfikować przyczyny jego zbyt niskiej skuteczności. W tabelach 2 i 3 podano przykłady kryteriów z obu wymienionych klas.

Tabela 2. Przykłady kryteriów zewnętrznych

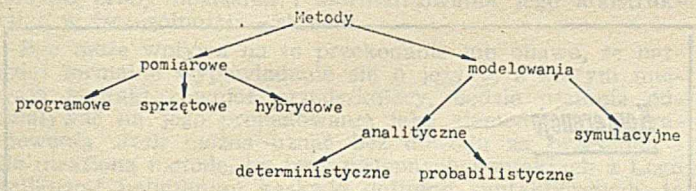
Kryterium	Definicja
Przepustowość	Ilość informacji przetworzona przez system w jednostce czasu dla danego obciążenia; używane są różne jednostki ilości informacji przetwarzanej przez system: praca: zadanie, program, proces, transakcja, interakcja, instrukcja; żadna z nich nie jest niezależna od systemu (organizacja, język wewnętrzny) i od obciążenia (skład, język programowania)
Przepustowość względna	Stosunek czasu niezbędnego do przetworzenia określonej ilości informacji przy określonym obciążeniu w systemie S_1 do czasu niezbędnego do przetworzenia tej samej ilości informacji przy tym samym obciążeniu systemie S_2
Przepustowość maksymalna	Maksymalna ilość informacji przetwarzana przez system w jednostce czasu dla określonego obciążenia
Czas odpowiedzi	Czas upływający między podaniem zadania na wejście systemu a uzyskaniem odpowiedzi; w systemie interakcyjnym każde polecenie z końcówki użytkownika jest traktowane jako nowe zadanie
Dostępność	Procent czasu, w którym system jest dostępny dla użytkownika

METODY OCENY DZIAŁANIA SYSTEMÓW KOMPUTEROWYCH

Na rysunku 1 przedstawiono klasyfikację metod, które pozwalają uzyskiwać informację o wartościach (w szczególności optymalnych) kryteriów oceny działania systemów komputerowych dla danego modelu systemu i jego obciążenia.

Tabela 3. Przykłady kryteriów wewnętrznych

Kryterium	Definicja
Współczynnik opóźnienia	Stosunek czasu odpowiedzi dla zadania do czasu jego przetwarzania
Czasowy współczynnik wieloprogramowości	Stosunek czasu odpowiedzi dla zadania w warunkach wieloprogramowości do czasu odpowiedzi dla tego zadania, gdy jest ono jedynym zadaniem w systemie
Współczynnik przyspieszenia	Stosunek czasu systemowego niezbędnego do wykonania danego zbioru zadań w warunkach wieloprogramowości do czasu systemowego niezbędnego do sekwencyjnego wykonania tego zbioru zadań (czas systemowy jest to czas, w którym co najmniej jeden moduł systemowy jest zajęty)
Współczynnik wykorzystania modułu (sprzętu, systemu operacyjnego, kompilatora itd.)	Stosunek czasu wykorzystania danego modułu w określonym przedziale czasu do długości tego przedziału
Współczynnik wykorzystania systemu	Suma ważona współczynników wykorzystania poszczególnych zasobów systemu
Narzut czasowy	Procent czasu procesora centralnego zużywany przez system operacyjny
Czas reakcji	Czas upływający między zakończeniem wprowadzania danych do systemu a otrzymaniem pierwszego kwantu czasu procesora
Częstotliwość błędów strony	Liczba błędów strony w jednostce czasu
Współczynnik jednoczesności pracy	Procent czasu, w którym dwa (lub więcej) składniki systemu pracują równocześnie



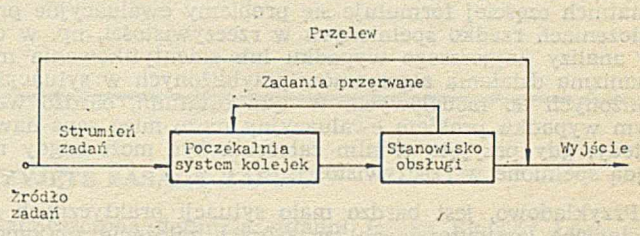
Rys. 1. Klasyfikacja metod oceny działania systemów komputerowych

W metodach pomiarowych informacja ta jest uzyskiwana bezpośrednio z systemu komputerowego, natomiast **w metodach modelowania** — z jego modelu. Model może być analityczny lub symulacyjny, przy czym zastosowanie metod numerycznych do uzyskiwania rozwiązań, nie przesądza o zaliczeniu metody do klasy metod symulacyjnych. O tych ostatnich mówi się tylko wtedy, gdy używa się modelu symulującego zachowanie się rzeczywistego systemu podczas jego działania.

Metody pomiarowe można — rzecz jasna — stosować tylko wtedy, gdy system w pewnej konfiguracji fizycznej już istnieje, a więc w problemach wyboru, a praktycznie głównie w problemach ulepszania. Natomiast w problemach projektowania, zwłaszcza na etapach wstępnych, mających istotne znaczenie dla efektu końcowego, można stosować jedynie metody modelowania. Warto także podkreślić głębokie podobieństwo koncepcyjne między metodami pomiarowymi a metodami symulacyjnymi, wynikające z faktu, że z założenia model symulacyjny ma działać w sposób zbliżony do rzeczywistego systemu.

Na rysunku 1 wyróżniono wśród metod analitycznych metody **deterministyczne** i **probabilistyczne**, co wymaga krótkiego komentarza. Po pierwsze, do deterministycznych zalicza się metody, w których nie jest wykorzystywany żaden parametr (rozumiany ogólnie — jak w teorii złożoności obliczeniowej problemów decyzyjnych) o opisie niedeterministycznym. W szczególności, można nie dysponować żadną aprioryczną wiedzą o danym parametrze; wiedza ta nie jest także uzyskiwana w sposób adaptacyjny. Na przykład, nie istnieje a priori żadna wiedza o momentach gotowości zadań — momenty te stają się znane w sposób dynamiczny. Po drugie, na rysunku 1 nie wymieniono metod, w których indeterminizm jest opisany inaczej niż w sposób probabilistyczny (np. na gruncie teorii zbiorów rozmytych), gdyż ich wykorzystanie do oceny działania systemów komputerowych jest, jak dotychczas, znikome.

Wśród metod probabilistycznych najlepiej opracowane matematycznie i najczęściej wykorzystywane w praktyce są metody, w których system komputerowy jest rozpatrywany jako system masowej obsługi. Metody te mają także najbogatszą literaturę, np. [6—12]. Dla porównania podstawowych założeń przyjmowanych w metodach probabilistycznych (ściślej — masowoobsługowych) i deterministycznych, należy przypomnieć krótko metodykę postępowania w wypadku tych pierwszych, posługując się najprostszym modelem, w którym zadania ubiegają się tylko o jeden zasób dostępny jako jedna jednostka. Jednostka ta (najczęściej procesor centralny) jest reprezentowana jako stanowisko obsługi w jednostanowiskowym systemie obsługi, którego ogólny schemat przedstawiono na rysunku 2.



Rys. 2. Jednostanowiskowy system obsługi

Dla takiego systemu zakłada się w ogólności, że znane są:
— wymiarowość źródła zadań (zgłoszeń),
— rozkład odstępu czasowego **U** między przybyciem kolejnych zadań,
— pojemność poczekalni (rozmiar odpowiedniego bufora),
— algorytm szeregowania (regulamin obsługi),
— rozkład czasu obsługi V.

Największe znaczenie praktyczne, z punktu widzenia oceny działania systemu, ma analiza w stanie równowagi statystycznej, w którym zostały osiągnięte granice:

$$\lim_{t\to\infty} P[N(t)=k]=p_k, \quad k=1,2,\cdots; \quad \lim_{t\to\infty} N(t)=N$$

gdzie N(t) jest liczbą zadań w systemie (w poczekalni i na stanowisku obsługi) w chwili t.

Warto podkreślić, że sam warunek dostateczny osiągania stanu równowagi statystycznej (w rozpatrywanym systemie ma on postać $\varrho=\lambda/\mu<1$, gdzie $\lambda=1/\overline{U}$, $\mu=1/\overline{V}$) ma bardzo istotne znaczenie i w bardziej złożonych wypadkach ogólna sieć kolejkowa jest głównym przedmiotem badań.

Przedmiotem analizy systemu w stanie równowagi statystycznej są w ogólności rozkłady zmiennych losowych: N, W (czas oczekiwania na obsługę) oraz T = W + V (czas odpowiedzi). W szczególności interesujące są wartości średnie $\overline{T}=\overline{W}+1/\mu$, $\overline{N}=\lambda\overline{T}$. Dla algorytmów szeregowania zapewniających preferencje zadaniom w funkcji ich długości czasu trwania obsługi bardzo istotna jest ponadto znajomość rozkładów warunkowych zmiennych losowych W i T, opisanych dystrybuantami:

$$F_W(x\,|\,t)=P(W<x\,|\,V=t)$$
$$F_T(x\,|\,t)=P(T<x\,|\,V=t)$$

a w szczególności — ich wartości średnich $\overline{W}(t)$ i $\overline{T}(t)$, gdzie $\overline{T}(t)=\overline{W}(t)+t$.

Poza gronem specjalistów dość rozpowszechniony jest pogląd, że metody deterministyczne przyjmują silniejsze założenia niż metody masowoobsługowe i wobec tego mają mniejsze znaczenie praktyczne. Źródło tego nieporozumienia leży w niezrozumieniu istoty badań ewaluacyjnych. Otóż badania te nie mają na celu wyeliminowania projektanta i zastąpienia go zautomatyzowanymi procedurami projektowania optymalnych systemów. W wypadku systemów komputerowych jest to aktualnie praktycznie niemożliwe, ze względu na hierarchiczną złożoność tego systemu, powiązań między jego podsystemami i wielorakość aspek-

tów, które należy uwzględnić. Intuicji projektanta (zespołu projektantów) nic zatem ostatecznie nie zastąpi, można ją tylko, i to niekiedy w znacznym stopniu, podbudować i ukierunkować. Temu właśnie celowi służą badania ewaluacyjne. Mają one to do siebie, że odpowiadają na pytania o działanie systemu lub jego fragmentów z punktu widzenia różnych kryteriów i przy różnych założeniach co do modelu systemu i jego obciążenia. W niebanalnych sytuacjach dopiero suma tych odpowiedzi pomaga projektantowi w wyborze najwłaściwszego rozwiązania. Natomiast poszczególne założenia są zwykle spełnione tylko w przybliżeniu, w odniesieniu do fragmentów systemu i jego obciążenia, oraz lokalnie w sensie czasowym. Dotyczy to także założeń czynionych w metodach probabilistycznych, tylko że świadomość tego faktu jest zwykle mniejsza (ze szkodą dla interpretacji wyników) niż w odniesieniu do metod deterministycznych. Może to wynikać stąd, że w tych ostatnich częściej formułuje się problemy ewaluacyjne przy założeniach rzadko spełnionych w rzeczywistości, np. w celu analizy najgorszego wypadku lub zidentyfikowania mechanizmu działania algorytmów przybliżonych w sytuacjach zbliżonych do modelowych. W tym ostatnim, bardzo ważnym wypadku problem ewaluacyjny może mieć sens nawet wtedy, gdy przyjęte w nim założenia być może nigdy nie będą spełnione w rzeczywistości.

Przykładowo, jest bardzo mało sytuacji praktycznych (w systemach komputerowych), w których założenie znajomości czasów wykonywania zadań, dosłownie rozumiane, jest spełnione. Jednak w deterministycznym problemie ewaluacyjnym jako te czasy mogą wystąpić górne granice rzeczywistych czasów wykonywania zadań lub wartości średnie tych czasów, traktowanych jako zmienne losowe.

Pierwsza sytuacja jest charakterystyczna dla problemów szeregowania zadań w środowisku sztywnego czasu rzeczywistego (ang. hard real time), gdy zadania muszą zostać wykonane przed upływem narzuconych terminów (linii krytycznych). Chcąc odpowiedzieć, na przykład, na pytanie o minimalną liczbę procesorów zapewniającą wykonanie danego zbioru zadań przy takich warunkach, trzeba rozwiązać odpowiedni deterministyczny problem szeregowania, w którym wystąpią górne granice czasów wykonywania zadań. Można je łatwo obliczyć, znając funkcje złożoności obliczeniowej czasowej odpowiednich algorytmów (zwykle są to algorytmy sterowania wykonywane dla cyklicznie pobieranych danych ze sterowanego procesu).

W drugiej sytuacji, rozwiązując deterministyczny problem minimalizacji długości uszeregowania, otrzymuje się optymistyczne oszacowanie wartości średniej czasu wykonania danego zbioru zadań [4].

Czasy wykonywania zadań mogą też być znane a posteriori, po ich wykonaniu; rozwiązanie odpowiedniego problemu szeregowania jest wówczas pomocne w ocenie aktualnie wykorzystywanej procedury szeregującej. Często zakłada się znajomość czasów wykonywania zadań, a nawet ich równość (czasy jednostkowe) po to, aby konstruując dokładny (czyli znajdujący dla każdych danych uszeregowanie optymalne, jeśli ono istnieje) algorytm szeregowania dla przyjętego kryterium, rozpoznać mechanizm szeregowania, który będzie dawał dobre wyniki przybliżone z punktu widzenia tegoż kryterium, gdy czasy wykonywania zadań nie są znane, nawet w sensie probabilistycznym. W ten sposób skonstruowano wiele efektywnych algorytmów szeregowania, z których liczne przebadano z punktu widzenia zachowania się zarówno średniego, jak i w najgorszym przypadku.

Ta metodyka konstruowania algorytmów, przybliżonych przez identyfikację mechanizmu działania algorytmów dokładnych, dotyczy oczywiście nie tylko szeregowania zadań, lecz jest powszechnie stosowana w badaniach ewaluacyjnych. Jej użyteczność jest szczególnie widoczna w wypadku złożonych modeli, obejmujących przywłaszczalne i nieprzywłaszczalne zasoby systemu, gdy zastosowanie metod masowoobsługowych jest istotnie ograniczone [2]. Metody deterministyczne stanowią podstawę dla tej metodyki, gdyż pozwalają formułować i rozwiązywać problemy, w których poszukuje się algorytmów dokładnych, przy bardzo różnorodnych założeniach dotyczących charakterystyk czasowych zadań, żądań zasobowych, ograniczeń kolejnościowych, kryteriów oceny działania systemu itp. [1, 3].

Podsumowanie tych kilku uwag prowadzi do stwierdzenia, że płytkie porównywanie metod deterministycznych i probabilistycznych jest nieporozumieniem. Obie klasy metod dotyczą różnych klas problemów ewaluacyjnych i zwykle dopiero ich łączne umiejętne wykorzystanie, w połączeniu z intuicją projektanta, może dać właściwy efekt.

LITERATURA

[1] Błażewicz J., Cellary W., Słowiński R., Węglarz J.: Badania operacyjne dla informatyków. WNT, Warszawa, 1983

[2] Cellary W.: Rozdział zasobów w systemach komputerowych — próby podejścia globalnego. Rozprawy, nr 127, Wydawnictwa Uczelniane Politechniki Poznańskiej, 1981

[3] Coffman E. G. Jr (red.): Teoria szeregowania zadań. WNT, Warszawa, 1980

[4] Coffman E. G. Jr, Denning P. J.: Operating Systems Theory. Prentice-Hall, Englewood Cliffs (NJ), 1973

[5] Ferrari D.: Computer Systems Performance Evaluation. Prentice-Hall, Englewood Cliffs (NJ), 1978

[6] Gelenbe E., Mitrani I.: Analysis and Synthesis of Computer Systems. Academic Press, New York, 1980

[7] Kleinrock L.: Queueing Systems. Vol. I — Theory, Vol. II — Computer Applications, Wiley, New York, 1975, 1976

[8] Kobayashi H.: Modelling and Analysis — An Introduction to System Performance Evaluation Methodology. Addison-Wesley, Reading (MA), 1978

[9] Lavenberg S. S. (ed.): Computer Performance Modeling Handbook. Academic Press, New York, 1983

[10] Lazowska E. D. et al.: Quantitative System Performance. Prentice Hall, Englewood Cliffs (NJ), 1984

[11] Morris M. F., Roth P. F.: Computer Performance Evaluation — Tools and Techniques for Effective Analysis. Van Norstrand Reinhold Company, New York, 1982

[12] Sauer G. H., Mani Chandy K.: Computer System Performance Modelling. Prentice-Hall, Englewood Cliffs (NJ), 1981

[13] Svobodova L.: Computer Performance Measurement and Evaluation Methods — Analysis and Applications. Elsevier, New York, 1976.

Konferencje

Systemy Zarządzania Bazami Danych

W dniach od 19 do 24 października 1987 r. w Cedzynie koło Kielc odbędzie się 10. Międzynarodowe Seminarium nt. **Systemów Zarządzania Bazami Danych** (10th International Seminar on Data Base Management Systems). Organizatorem konferencji jest Centrum Projektowania i Zastosowań Informatyki (CPiZI) ZETO-ZOWAR, Warszawa.

Program konferencji obejmuje następującą tematykę:

- systemy zarządzania bazami danych,
- metodologie i narzędzia do projektowania baz danych,
- rozproszone bazy danych,
- maszyny baz danych,
- sprawność działania,
- ochrona danych,
- systemy z bazą wiedzy,
- zastosowania.

Oprócz prac nadesłanych na konferencję będą także prezentowane prace przygotowane na zamówienie organizatorów.

Bliższych informacji udziela Ewa Giera, sekretarz komitetu programowego, CPiZI ZETO-ZOWAR, al. Niepodległości 190, 00-608 Warszawa, tel. 41-21-81, telex 813492 CPZI pl.

EWA GURBIEL
HELENA KRUPICKA
ZDZISŁAW PŁOSKI
Wrocław

Składnia języka Logo — próba formalizacji (I)

Język Logo, łatwy w pierwszych kontaktach, mile zdumiewający nowicjuszy prostymi do wyrażenia, a ciekawymi wizualnie efektami graficznymi, jest bądź winien być — co się zwykle podkreśla w wypowiedziach na jego temat — uniwersalnym językiem programowania o statusie użytkowym nie niższym niż Basic (który przewyższa strukturalnymi koncepcjami programowania).

Jednakże, deklaracjom o uniwersalności i „stateczności" Logo brakuje w praktyce istotnego elementu. Nigdy dotąd (przynajmniej w krajowym środowisku informatycznym i w źródłach dostępnych autorom artykułu), nie ważono się mówić o Logo profesjonalnym językiem informatyki, nie podjęto próby nazwania i sformalizowania jego konstrukcji, a w szczególności — składni.

Być może wpływa na to przekonanie lub obawa, że bardziej formalne wypowiadanie się o języku, z którym miewają kontakt również przedszkolacy, będzie ujemnie oddziaływać na jego propagowanie jako elementarza programowania. Jeśli można uznać bez dowodu za przesadną i źle utrafioną metodę, by przy wstępnych kontaktach z Logo aplikować nauczanym kaskadę definicji syntaktycznych, to jednak już pierwsze doświadczenia z nauczania Logo uczniów szkoły średniej (eksperyment nauczania Logo jest prowadzony przez autorów od września 1985 roku) w ostry sposób uwidaczniają niedostatek środków wypowiedzi o języku Logo, zwłaszcza tam, gdzie należy rozstrzygać o składniowych i znaczeniowych szczegółach, a potrzeby takie powstają nadspodziewanie szybko. Pozostawanie w takiej sytuacji ze stwierdzeniami, że *„tak* trzeba (napisać)" lub „użyj *tego* w tym miejscu", jest wielce niezadowalające zarówno ze względu na złą propagandę informatyki (a języków programowania w szczególności), jako dziedziny, w której wolno dopuszczać wątpliwy luksus wyrażeń nieprecyzyjnych, jak również ze względów ogólnodydaktycznych (nieporządek w myśleniu i wypowiedziach przenoszony w nowe środowisko techniczne).

Aby Logo mogło się stać pełnowartościowym językiem programowania (czego ze wszech miar jest warte), powinno doczekać się opracowania formalnego, które przyniesie w szczególności aparat pojęciowy, pozwalający na możliwie ścisłe mówienie i wymianę informacji (jednoznaczne wzajemne rozumienie się) o tym języku.

Niniejsze opracowanie jest próbą takiego formalnego potraktowania Logo. Przytoczona definicja składni i zawarte w niej pojęcia syntaktycznych klas w Logo niosą propozycje formułowania wypowiedzi o konstrukcjach tego języka. Nie traktujemy ich jako ostateczne. Przeciwnie — oczekujemy dyskusji (zarówno w sferze obranego słownictwa, jak też odnośnie przyjętej klasyfikacji).

Prezentowana składnia dotyczy dialektu Sinclair Logo. Realizacja ta dostarcza dodatkowego, gorzkiego argumentu co do zasadności formalnego definiowania składni i precyzyjnego definiowania semantyki języka programowania w jego zaraniu, nawet gdyby to miał być język programowania dla przysłowiowych przedszkolaków. Oprócz nich istnieją nauczyciele, którym jest właśnie potrzebny możliwie doskonały system relacyjno-pojęciowy do wykładania, a później — weryfikowania danego języka. Wspomnianej goryczy doświadczy — jak stało się to naszym udziałem — każdy, kto postulat równorzędności Sinclair Logo wobec innych języków, jako niezawodnego i jednoznacznie określonego środka programowania, zechce sprawdzić w praktyce.

PRZYJĘTE ZASADY ZAPISU

Zamieszczona definicja składni języka Sinclair Logo nie jest w zupełności sformalizowana. Pełna formalizacja byłaby ryzykowna, ze względu na wtórne pochodzenie prezentowanych równań syntaktycznych (firmowy podręcznik Sinclair Logo wspomina o gramatyce Logo, lecz jej nie zawiera), a przede wszystkim z powodu osobliwości interpretera języka.

W definicjach przyjęto następujące zasady typograficzne:

1. Każda definicja ma postać:
nazwa-klasy-syntaktycznej:
człon-definiujący

2. Elementy metajęzyka są wydrukowane kursywą. W nazwach wielowyrazowych używa się myślnika, np.: *liczba-dziesiętna*. Element z umieszczonym bezpośrednio po nim wielokropkiem oznacza możliwość występowania w danym miejscu dowolnego, niepustego ciągu takich samych elementów np.: *cyfra...* reprezentuje ciągi: *cyfra, cyfra cyfra, cyfra cyfra cyfra,* itd.

3. Alternatywne człony w definicjach syntaktycznych zapisuje się od nowego wiersza, np.:
liczba-całkowita:
cyfra...
— *cyfra...*

Tam, gdzie członów alternatywnych jest wiele, stosuje się wyliczanie ich w kilku kolumnach.

4. Elementy języka Logo są wydrukowane prostą czcionką, bezszeryfową. Ponieważ małe litery w nazwach procedur i funkcji są utożsamiane z wielkimi, a niektóre nazwy mają równoważne im skróty, przyjmuje się, że:
— litery tworzące skrót są pisane w nazwie jako wielkie, pozostałe — jako małe, np.: PRint, ForwarD;
— nazwa nie mająca skrótu jest zapisana wielkimi literami, np.: MAKE, REPEAT.

5. Miejsca obowiązkowego użycia znaku ⟨ENTER⟩ zaznaczono explicite. Miejsc, w których obowiązkowe bądź zakazane są odstępy, opis formalny nie uwidacznia — określają je uwagi następujące niżej.

6. Linie poziome oddzielają graficznie jednotematyczne kompleksy składniowe.

Pomiędzy dowolnymi dwoma reprezentantami dowolnej klasy syntaktycznej, jeśli nie rozdziela ich znak ⟨ENTER⟩, jest wymagany w języku Logo co najmniej jeden odstęp. Wyjątkowo, nie jest konieczne wydzielanie odstępami ograniczników oraz nawiasów kwadratowych, a ponadto w słowach Logo (w szczególności w liczbach i nazwach) stosowanie odstępów jest niedozwolone. Nie należy mylić odstępu użytego w jego naturalnej w Logo funkcji separatora z odstępem literalnym, poprzedzonym znakiem ukośnika (ang. backslash), który zastąpiono w tekście kreską pionową, ze względu na brak ukośnika w zestawie znaków drukarni.

Znak ⟨ENTER⟩ pełni separującą rolę odstępu. W wypadku wystąpienia wewnątrz instrukcji, znak ⟨ENTER⟩ przerywa jej kompletowanie, powodując automatyczne domknięcie wszystkich ewentualnych nawiasów. Należy zatem zważać na konsekwencje użycia znaku ⟨ENTER⟩, zwłaszcza wewnątrz wyrażeń listowych (instrukcja REPEAT i podobne).

Niektóre definicje uzupełniono opisami semantycznymi. Gdzie indziej — w celu przybliżenia znaczeń — posłużono się pomocniczymi nazwami metajęzykowymi, zbędnymi w opisie składni, np.: *nazwa-funkcji, odległość, kąt* itp.

Zastosowano zstępującą kolejność definiowania — od określenia pojęcia *opisu* do sklasyfikowania znaków wchodzących w skład alfabetu języka.

POJĘCIE PROGRAMU — SKŁADNIA OPISÓW I INSTRUKCJI

Jeśli pojmować termin „program" w kategoriach czysto składniowych, to w takim rozumieniu w języku Logo pojęcie programu nie występuje. Rozważanie składni wypada tu rozpocząć od stwierdzenia, że w języku Logo programuje się za pomocą opisów procedur lub funkcji, instrukcji i wyrażeń. Zestawy konstruowanych w Logo procedur można wykorzystywać — stosownie do potrzeb — zupełnie dowolnie, od wywoływania ich w sposób wzajemnie od siebie niezależny, na zespołach monolitycznych pod względem sposobu przekazywania sterowania kończąc. Przez wzgląd na tę ostatnią możliwość, całkowite uniknięcie posługiwania się potocznym terminem „program", przy programowaniu w Logo, jest trudne. Przyznajmy, że nawet — niekonieczne.

opis:
  opis-procedury
  opis-funkcji
opis-procedury:
  nagłówek-procedury treść-procedury
nagłówek-procedury:
  TO *nazwa-procedury* ⟨ENTER⟩
  TO *nazwa-procedury parametr-formalny...* ⟨ENTER⟩
nazwa-procedury:

Nazwą procedury może być niepuste słowo, które nie rozpoczyna się od znaków: ", :, |", |:, jest różne od liczby, ogranicznika i znaków literalnych postaci |*ogranicznik* oraz wszystkich nazw procedur pierwotnych i funkcji pierwotnych. Nazwy procedur pierwotnych występują w wykazie instrukcji pierwotnych; funkcje pierwotne wymieniono w punkcie: składnia wyrażeń.

parametr-formalny:
  : symbol...

Po znaku : nie może wystąpić odstęp.

treść-procedury:
  instrukcja... (ENTER) END
  END
instrukcja:
  instrukcja-pusta
  instrukcja-procedury
  (instrukcja)
instrukcja-pusta:

Instrukcja pusta nie jest reprezentowana przez żaden znak.

instrukcja-procedury:
  instrukcja-pierwotna
  nazwa-procedury
  nazwa-procedury parametr-aktualny...
paramer-aktualny:
  wyrażenie

SKŁADNIA PROCEDUR PIERWOTNYCH

Język Logo charakteryzuje duża liczba procedur i funkcji pierwotnych. W tym punkcie wylicza się procedury pierwotne wersji Sinclair Logo, grupując je wedle ich znaczeń. Funkcje pierwotne są prezentowane w dalszej części artykułu, przy opisie wyrażeń.

instrukcja-pierwotna:
  instrukcja-przypisania
  instrukcja-powtarzania
  instrukcja-warunkowa
  instrukcja-powrotu
  instrukcja-czekania
  instrukcja-interpretowania-listy

Wyliczone dotąd instrukcje, razem z wywołaniami procedur zdefiniowanych przez użytkownika, tworzą klasę instrukcji organizacji sterowania w programie.

  instrukcja-graficzna
  instrukcja-tekstowa
  instrukcja-dźwiękowa

Instrukcje tych grup służą do bezpośredniej komunikacji między programem a użytkownikiem.

  instrukcja-doglądania
  instrukcja-redagowania
  instrukcja-usuwania
  instrukcja-kopiowania
  instrukcja-generowania
  instrukcja-drukarkowa
  instrukcja-magnetofonowa
  instrukcja-egzotyczna

Wśród procedur pierwotnych nie ma instrukcji czytania danych z klawiatury komputera. Zadanie to wypełniają odpowiednie funkcje Logo.

instrukcja-przypisania:

  MAKE *wyrażenie-słowowe wyrażenie*

Wartością wyrażenia słowowego powinna być *nazwa zmiennej* (p. punkt: składnia wyrażeń).

instrukcja-powtarzania:
  REPEAT *wyrażenie-arytmetyczne ciąg-instrukcji*
instrukcja-warunkowa:
  IF *wyrażenie-logiczne ciąg-instrukcji*
  IF *wyrażenie-logiczne ciąg-instrukcji ciąg-instrukcji*
instrukcja-powrotu:
  BYE
  TOPLEVEL
  STOP
  OutPut *wyrażenie*
instrukcja-czekania:
  WAIT *wyrażenie-arytmetyczne*
instrukcja-interpretowania-listy
  RUN *ciąg-instrukcji*

Ciąg instrukcji jest *wyrażeniem* listowym.

instrukcja-graficzna:

ForwarD *odległość*	
RighT *kąt*	HideTurtle
LefT *kąt*	ShowTurtle
BacK *odległość*	SETX *współrzędna-punkt*
DOT *współrzędne-punktu*	SETY *współrzędna-punkt*
HOME	SETPOS *współrzędne-pu*
CLEAN	SETHeading *kąt*
ClearScreen	SETBG *kolor*
PenUp	SETBordeR *kolor*
PenDown	SETPC *kolor*
PENREVERSE (w skrócie: PX)	SETSCRUNCH *proporcja*
PenErase	WINDOW
	FENCE
	WRAP

Odległość, kąt, współrzędna punktu oraz *kolor* mają składnię *wyrażenia arytmetycznego. Współrzędne punktu* i *proporcja* są określane za pomocą *wyrażenia listowego* (p. punkt: składnia wyrażeń).

instrukcja-tekstowa:

TextScreen	FLASH
SETTC *kolory*	INVERSE
SETCURsor *współrzędne-znaku*	NORMAL
BRIGHT *wyrażenie-arytmetyczne*	ClearText
OVER *wyrażenie-arytmetyczne*	*instrukcja-drukowania*

Współrzędne znaku oraz *kolory* są określane za pomocą *wyrażenia listowego.* Powyższe instrukcje ustalają sposób uwidaczniania tekstu na ekranie. *Instrukcje drukowania* wykonują przesyłanie tekstu na ekran.

instrukcja-drukowania:
  PRint *wyrażenie*
    (PRint *wyrażenie...*)
  TYPE *wyrażenie*
    (TYPE *wyrażenie...*)
  SHOW wyrażenie
instrukcja-dźwiękowa:
  SOUND *nuta*

Nuta, czyli określenie czasu trwania i wysokości tonu, jest *wyrażeniem listowym.*

Drukowaniem na ekranie opisów procedur i innych informacji przechowywanych w pamięci zajmują się instrukcje nazwane tu wspólnie *instrukcjami doglądania.*

instrukcja-doglądania:

POTS	POALL
PO *wyrażenie*	.CONTENTS
POPS	
PONS	.PRIMITIVES

instrukcja-redagowania:
EDit
EDit *przedmiot-redagowania*
EDNS
przedmiot-redagowania:
[]
nazwa-procedury
" nazwa-procedury
: nazwa-procedury
[nazwa-procedury...]

Po znakach " i : nie może wystąpić odstęp. Składnia parametru aktualnego instrukcji redagowania objawia jedną z niekonsekwencji interpretera Sinclair Logo.

instrukcja-usuwania
ERase *wyrażenie*
ERN *wyrażenie*
ERNS
ERPS
ERALL
RECYCLE
instrukcja-kopiowania:
COPYDEF *wyrażenie-słowowe wyrażenie-słowowe*
instrukcja-generowania:
DEFINE *wyrażenie-słowowe wyrażenie-listowe*
instrukcja-drukarkowa:
PRINTON
PRINTOFF
COPYSCREEN
instrukcja-magnetofonowa:
instrukcja-zapisu
instrukcja-odczytu
instrukcja zapisu:
SAVED *nazwa-pliku*
SAVESCR *nazwa-pliku*
SAVE *nazwa-pliku wyrażenie*
SAVEALL
instrukcja-odczytu:
LOADD *nazwa-pliku*
LOADSCR *nazwa-pliku*
LOAD *nazwa-pliku*

W instrukcjach zapisu i instrukcjach odczytu *nazwy plików* są *wyrażeniami słowowymi.*

Pozostała grupa instrukcji dotyczy bądź współpracy programu w Logo z modułami (podprogramami lub blokami danych) przygotowanymi w kodzie wewnętrznym Spectrum, bądź komunikacji z rzadkimi na naszym rynku urządzeniami zewnętrznymi. Instrukcję .DEPOSIT warto porównać z funkcją .EXAMINE, a instrukcję .SERIALOUT z funkcją .SERIALIN.

instrukcja-egzotyczna:
instrukcja-bajtowa
instrukcja-podprogramu
instrukcja-microdrive'u
instrukcja-sieciowa
instrukcja-robotowa
strukcja-bajtowa:
.RESERVE *liczba-bajtów*
.DEPOSIT *adres-bajtu bajt*
.BLOAD *nazwa-pliku adres-bajtu*
.BSAVE *nazwa-pliku określenie-obszaru-bajtów*
instrukcja-podprogramu:
.CALL *adres-bajtu*
instrukcja-sieciowa:
.SETSERIAL *wyrażenie-arytmetyczne*
.SERIALOUT *bajt*
instrukcja-microdrive'u:
SETDRIVE *numer*
CATALOG
ERASEFILE *nazwa-pliku*
instrukcja-robotowa:
STARTROBOT
STOPROBOT

Bajt, adres bajtu, liczba bajtów i *numer* są *wyrażeniami arytmetycznymi. Nazwa-pliku* jest *wyrażeniem słowowym. Określenie obszaru bajtów* jest *wyrażeniem listowym.*

*

Uwagi doc. dra hab. Stanisława Waligórskiego z Instytutu Informatyki Uniwersytetu Warszawskiego przyczyniły się do przejrzenia pierwszej wersji niniejszej gramatyki i wprowadzenia w niej wielu zmian. Autorzy składają Mu za to podziękowania.

LITERATUTA

[1] Ross P.: Logo Programming. Addison-Wesley, Reading (MA), 1983.
[2] Sparer E.: Sinclair Logo 1 — Turtle graphic. Sinclair Research, Cambridge 1984.
[3] Sparer E.: Sinclair Logo 2 — Programming Reference Manual. Sinclair Research, Cambridge 1984.

W skrócie • W skrócie • W skrócie • W skrócie • W skrócie • W skrócie

■ Firma IBM wprowadza na rynek nowy model mikrokomputera osobistego, należącego do nowej klasy mikrokomputerów podręcznych, zwanych po angielsku laptop. Nowy mikrokomputer IBM PC Convertible jest wyposażony w ciekłokrystaliczny ekran o 24 wierszach po 80 znaków w wierszu i dwie stacje dyskietek 3,5-calowych o pojemności 720 KB każda. W komputerze jest zainstalowany procesor CMOS 8088 oraz statyczna pamięć CMOS RAM o pojemności 256 KB, z możliwością rozszerzenia o dalsze 512 KB. Klawiatura składa się z 78 klawiszy (w układzie zgodnym z poprzednimi modelami IBM PC) z dodatkowym klawiszem funkcyjnym do przesuwania kursora. Do zasilania używane są ładowalne baterie pozwalające na 6—10-godzinną pracę. Możliwe jest też zainstalowanie zasilacza. Komputer waży ok. 5,5 kg i ma wymiary 38,1×30,5×7,6 cm.

Podstawowa wersja kosztuje obecnie 1995 dolarów. Każde dodatkowe 128 KB pamięci kosztuje 195 dolarów. Zamiast ekranu ciekłokrystalicznego można dołączyć kartę graficzną (za 350 dolarów) i 9-calowy monitor monochromatyczny (za dalsze 180 dolarów) o rozdzielczości 640×200 pikseli. Możliwe jest też dołączenie 13-calowego monitora RGB (za 400 dolarów) o rozdzielczości 320×200 pikseli. Rozbudowując konfigurację komputera można dokupić kartę łącza równoległo-szeregowego (za 195 dol.) oraz mozaikową drukarkę termiczną (za 295 dol.). W celu zapewnienia przenośności oprogramowania do (z) komputerów IBM PC oraz XT i AT można do nich dokupić zewnętrzną stację dyskietek 3,5-calowych (za 395 dol.) lub stację montowaną wewnątrz IBM PC/XT (za 190 dol.).

Oprogramowanie komputera oparto na nowo wprowadzonym systemie operacyjnym PC-DOS w wersji 3.2. Nowy SystemApps zawiera pakiety planowania, podręcznego notesu, spisu telefonów oraz kalkulatora. Równocześnie z zaanonsowaniem przez IBM nowego komputera inne firmy produkujące oprogramowanie zaczęły przystosowywać swoje produkty do nowego typu dyskietek. Na przykład, znana firma Ashton-Tate reklamuje już Framework II, dBase III Plus, MultiMate Profesional Word Processor. Podobnie firma Lotus Development Corp. przystosowała już swoje pakiety 1—2—3 i Symphony. (WI)

*

■ Firma Alsys rozpoczęła reklamę najnowszego swojego produktu — kompilatora pełnego języka Ada dla mikrokomputera IBM PC/AT. Kompilator jest na stałe zapisany w 4-MBajtowej pamięci, sprzedawanej jako osobna karta. Program po kompilacji może zajmować do 16 MB pamięci. Cena kompilatora, czyli karty z pamięcią ROM wynosi 3000 dolarów. (WI)

*

■ Firma GTE Microcomponents, z Temple w stanie Arizona, oferuje nowe sterowniki terminali sieciowych, które zmniejszają pobieraną moc i objętość konstrukcji. Jest to mikroprocesor, który obsługuje modem i ma większe możliwości wejścia-wyjścia. Zawiera on również sprzęg telekomunikacyjny z licznikami wierszy i kolumn, generatorami, 27 liniami wejścia-wyjścia do klawiatury i inne układy. W układzie można programować syntezę częstotliwości, co pozwala stosować go w różnych standardowych systemach łączności, jak teletekst, przesyłanie danych FSK z prędkością 1200 bodów i inne. (JR)

W skrócie • W skrócie • W skrócie • W skrócie • W skrócie • W skrócie

ROBERT KOWALSKI
Imperial College of Science and Technology
Uniwersytet Londyński
Wielka Brytania

Programowanie logiczne i komputery nowej generacji (2)

W drugiej części artykułu przedstawiono przykład zastosowania programowania logicznego w prawodawstwie i omówiono inne związki dziedziny sztucznej inteligencji z inżynierią oprogramowania.

ZASTOSOWANIE W PRAWODAWSTWIE

Prawodawstwo jest szczególnie ciekawą dziedziną zastosowań, wykazującą, że wykonywalna analiza może być na tyle efektywna, iż żaden z końcowych etapów konwencjonalnego procesu produkcyjnego oprogramowania nie jest potrzebny. Z drugiej strony, prawodawstwo zdecydowanie nie jest dziedziną łatwą, gdyż wymaga złożonych metod reprezentacji wiedzy i wnioskowania. Jest bardziej skomplikowane niż, na przykład, tak typowe zastosowania sztucznej inteligencji jak rozumienie opowiadań dziecięcych czy diagnozowanie uszkodzeń. Tak więc, pod pewnymi, choć nie wszystkimi względami prawodawstwo jest trudniejszą dziedziną zastosowań.

Dziedzina sztucznej inteligencji obfituje w problemy wieloznaczności. Szczególnie często stykają się z nimi badacze języka naturalnego. W przepisach prawnych też mogą wystąpić niejednoznaczności, choć intencją prawodawcy nie jest (a w każdym razie nie powinno być) ich wprowadzanie. Dlatego też prawodawstwo jest idealną dziedziną do zmagania się z trudnymi zagadnieniami reprezentacji wiedzy i rozwiązywania problemów bez niebezpieczeństwa odejścia od zasadniczego tematu.

Formalizacja prawodawstwa ilustruje ponadto przyrostową metodę wytwarzania oprogramowania, zwaną **metodą prób i błędów.** Gdyby ktoś pisał metodą prób i błędów programy reprezentujące znaczenie przepisów prawnych, to byłby złym programistą. Dobrzy programiści zaczynają od ścisłych, co najmniej formalnych specyfikacji programów, a następnie od razu poprawnie je implementują. Tak więc, programista Prologu, który bez przerwy poprawia błędy w swoich programach, jest złym programistą. Natomiast dla osoby, która używa Prologu nie jako języka programowania, nawet nie jako języka formalnej specyfikacji, lecz jako języka służącego do analizowania wiedzy kryjącej się za wymaganiami użytkownika, próby i błędy są nieuniknione. Nawet matematycy dowodzą twierdzeń i rozwijają teorie aksjomatyczne metodą prób i błędów.

Jednakże formalizacja reguł i przepisów ma znacznie szersze zastosowanie. Stosuje się ją wszędzie tam, gdzie jakaś organizacja wykorzystuje ustalone zasady do prowadzenia swej działalności, niezależnie od tego czy mają one moc prawną czy też nie.

Jaką rolę odgrywają przepisy prawne? Istnienie reguł oznacza tyle, że nie trzeba każdego problemu rozwiązywać tak, jakby przed nim nie było podobnych. Daje to możliwość zastosowania reguł ogólnych i traktowania różnych klientów jednakowo, według tych samych kryteriów.

W istocie cała koncepcja reprezentacji wiedzy opartej na logice ma ważne implikacje. Podczas wydobywania wiedzy ekspertów i formułowania jej w postaci reguł, odkrywa się — często po raz pierwszy — jakie w rzeczywistości są te reguły. Proces wydobywania wiedzy może być trudny. Czasami nie można zorientować się, co ekspert myśli i w co wierzy. Ale dotyczy to nie tylko ekspertów. Trudno jest zorientować się, jakich reguł my sami używamy do rozwiązywania codziennych problemów. Gdyby można je wysłowić, wówczas udałoby się je zbadać, poddać krytyce i poprawić. Nawet gdyby pierwsze próby wysłowienia reguł były błędne, można by poprawić je metodą prób i błędów. Można by także zobaczyć, co one sobą reprezentują, sprawdzić je, określić, czy są rzetelne, czy dają się zastosować w ten sam sposób do każdego klienta i do nas samych.

Formalizacja reguł i przepisów ilustruje także możliwość odmiennego wykorzystania techniki systemów ekspertowych, polegającego nie tylko na zastosowaniu znanej wiedzy doświadczalnej czy egzekwowaniu prawa w indywidualnych przypadkach. Stanowi ilustrację, w jaki sposób formułowanie przepisów metodą prób i błędów może stać się ważnym narzędziem rozwijania i doskonalenia naszej wiedzy specjalistycznej (na przykład, gdy w danym przedsiębiorstwie nie ma eksperta od regulacji płac). Jedną z recept na dobry początek jest przyjęcie kilku hipotetycznych reguł. Zamiast sprawdzania ich na ludziach, testuje się je przy użyciu komputera metodą interakcyjną, opartą na wspólnym dla maszyny i człowieka modelu dedukcyjnego rozwiązywania problemów.

Rozważymy kilka przykładów z Brytyjskiego Aktu Narodowości, aby zobaczyć, na ile potwierdzają one teorię. Pierwszy paragraf Aktu dotyczy **uzyskania obywatelstwa na mocy urodzenia: Osoba urodzona w Zjednoczonym Królestwie podczas obowiązywania Aktu jest obywatelem brytyjskim, jeżeli w chwili jej urodzenia jej ojciec lub matka jest:**

(a) obywatelem brytyjskim lub ...

Warto zwrócić uwagę, że słowo „jeżeli" pojawia się w tekście niemal dokładnie tam, gdzie pojawiłoby się w reprezentacji regułowej.

Wnioskiem w tej pierwszej klauzuli Brytyjskiego Aktu Narodowości jest stwierdzenie, że osoba jest obywatelem brytyjskim. Istnieją jednak pewne warunki logiczne ukryte w składni wniosku. Pierwszy warunek mówi, że osoba powinna być urodzona w Zjednoczonym Królestwie, drugi zaś, że powinna być urodzona po wprowadzeniu Aktu w życie. Pozostałe warunki występują jawnie po słowie „jeżeli". Oczywiście, reprezentacja regułowa umożliwia bardzo naturalny sposób przedstawienia tego rodzaju wiedzy:

x jest obywatelem brytyjskim
jeżeli x urodził się w Zjednoczonym Królestwie
i x urodził się dnia y
i y jest dniem obowiązywania Aktu
i z jest rodzicem x
i z był obywatelem brytyjskim dnia y.

Autor wraz ze współpracownikami [1]) badał reprezentację Brytyjskiego Aktu Narodowości w Prologu. Około 80% z siedemdziesięciu stron Aktu zapisano w Prologu. Ponieważ wystąpiły trudności z uchwyceniem struktury Aktu, próbowano zastosować diagramy przepływu danych. Dość szybko okazało się, że ich użycie było niewłaściwe i to z dwóch powodów. Po pierwsze, wymagały one oznaczenia kierunku przepływu danych, co podobnie jak w przykładzie z klientem i produktami niepotrzebnie ograniczało różne możli-

[1]) Zespół współpracowników tworzyli: Marek Sergot, Fariba Sadri, Frank Kriwaczek i Peter Hammond [9].

wości zastosowania reguły. Po drugie, nie jest łatwo przedstawić związki logiczne między różnymi procesami na diagramie. Ostatecznie użyto grafów „i-lub", będących takimi diagramami przepływu danych, w których związki logiczne między procesami są jawne, natomiast przepływy pominięte.

Graf „i-lub" umożliwił ogólne spojrzenie na strukturę Aktu, ale niewiele pomógł, jeżeli chodzi o szczegóły reprezentacji wiedzy. Ponadto wkrótce stało się jasne, że praktycznie nie ma innych możliwości poza udoskonalaniem reguł metodą prób i błędów. W szczególności, nieodpowiedniość pierwszej próby formalizacji paragrafu 1.1a wyszła na jaw dopiero przy paragrafie 2.1a, dotyczącym **uzyskania obywatelstwa na mocy pochodzenia:**

Osoba urodzona poza Zjednoczonym Królestwem w czasie obowiązywania Aktu jest obywatelem brytyjskim, jeżeli w chwili urodzenia jej ojciec lub matka:
(a) jest obywatelem brytyjskim nie na mocy pochodzenia lub ...

Warto zwrócić uwagę na warunek „jest obywatelem brytyjskim nie na mocy pochodzenia". Oznacza on, że nasze wcześniejsze spostrzeżenie, iż wniosek z paragrafu 1.1a jest:

x jest obywatelem brytyjskim

było naiwne. Ponadto ignoruje on ukryte założenie mówiące, że x uzyskuje obywatelstwo w chwili urodzenia. Biorąc pod uwagę obydwa przeoczenia, można poprawić oryginalne sformułowanie, otrzymując kolejne przybliżenie:

x uzyskuje obywatelstwo brytyjskie na mocy paragrafu 1.1a dnia y
jeżeli x urodził się w Zjednoczonym Królestwie
i x urodził się dnia y
i y jest dniem obowiązywania Aktu
i z jest rodzicem x
i z jest obywatelem brytyjskim dnia y na mocy prawa w.

Podobnie można zdefiniować paragraf 2.1a:

x uzyskuje obywatelstwo brytyjskie na mocy paragrafu 2.1a dnia y
jeżeli x urodził się poza Zjednoczonym Królestwem
i x urodził się dnia y
i y jest dniem obowiązywania Aktu
i z jest rodzicem x
i z jest obywatelem brytyjskim dnia y na mocy prawa v
i v nie jest prawem na mocy pochodzenia.

W obu tych regułach istnieje jednak niezgodność między sposobami wyrażenia obywatelstwa we wniosku i w warunkach. Poza tym jest potrzebna jeszcze jedna reguła, która nie jest jawnie zapisana w Akcie, ale jest zupełnie oczywista:

x jest obywatelem brytyjskim na mocy prawa dnia y
jeżeli x uzyskał obywatelstwo brytyjskie dnia z na mocy prawa w
i y jest po dacie z
i x żyje dnia y
i x nie wyrzekł się obywatelstwa brytyjskiego przed dniem y
i x nie został pozbawiony obywatelstwa brytyjskiego przed dniem y.

Inaczej mówiąc, osoba jest obywatelem brytyjskim określonego rodzaju i danego dnia, o ile wcześniej uzyskała to obywatelstwo, żyje, nie wyrzekła się go ani nie została go pozbawiona.

Paragraf 48 dotyczy mniej oczywistej sytuacji, kiedy osoba może być uznana za obywatela brytyjskiego po śmierci. Ojciec, który nie żyje w chwili urodzenia swego dziecka, jest uważany za obywatela brytyjskiego w chwili jego urodzenia, o ile był on obywatelem brytyjskim w chwili swej śmierci.

Reguły te ilustrują część najwyższego poziomu Brytyjskiego Aktu Narodowości. Warunki występujące w tych i inych regułach mogą być spełnione na wiele sposobów.

Niektóre warunki mogą być **zdefiniowane przez reguły.** Na przykład, warunek:

z osiadł w Zjednoczonym Królestwie dnia y

pochodzący z paragarafu 1.1b, jest określony w paragrafach 50.2, 50.3, 50.4 i jego definicja jest w naturalny sposób reprezentowana za pomocą reguł.

Inne warunki mogą być **zdefiniowane przez dane.** Na przykład, warunek:

z jest terytorium zależnym od Wielkiej Brytanii

jest zdefiniowany przez listę terytoriów wyliczonych w zestawieniu 6. Z każdym terytorium jest związane określone stwierdzenie, np.:

Gibraltar jest terytorium zależnym od Wielkiej Brytanii.

Każde takie stwierdzenie może być uważane za banalną regułę, mającą jeden wniosek i żadnych warunków.

Warunki mogą być też **obliczane przez programy,** np.:

y jest dniem obowiązywania Aktu.

Każdy program jest procedurą lub zbiorem procedur, które mogą być reprezentowane przez reguły, a te z kolei mogą być stosowane wstecz w celu zredukowania problemów do podproblemów.

Warunki mogą być spełnione także przez **zadawanie pytań użytkownikowi,** np.:

x urodził się dnia y.

Zazwyczaj każdy warunek może być rozstrzygnięty albo przez komputer, albo przez użytkownika. Komputer może zorientować się, że nie potrafi rozwiązać danego problemu i wówczas może automatycznie prosić użytkownika o rozwiązanie.

Warunki mogą być również spełnione przez zadawanie pytań ekspertowi. Na przykład, warunek:

x zamieszkuje w Zjednoczonym Królestwie w dniu y

nie jest zdefiniowany w Akcie, a decyduje o nim Sekretarz Stanu. Gdyby Sekretarz Stanu był nieobecny, należałoby skonsultować się z ekspertem lub systemem ekspertowym.

Oparta na logice formalizacja Brytyjskiego Aktu Narodowości, uzyskana metodą prób i błędów, jest przykładem użycia techniki programowania logicznego w zastosowaniu do dziedziny mającej cechy zarówno konwencjonalne, jak i nowatorskie. Z jednej strony, jeżeli ograniczyć się do problemów określania obywatelstwa, technika ta nie różni się bardzo od skomplikowanego zastosowania z dziedziny przetwarzania danych. Z drugiej strony, przy odpowiedniej maszynie wnioskującej ta sama reprezentacja może być, przynajmniej teoretycznie, użyta do generowania i testowania dowolnych konsekwencji logicznych Aktu. W obu wypadkach skrócono konwencjonalny cykl produkcyjny oprogramowania, doprowadzając go do końca bez wychodzenia poza fazę analizy systemu.

INNE ZWIĄZKI SZTUCZNEJ INTELIGENCJI Z INŻYNIERIĄ OPROGRAMOWANIA

Do tej pory zwrócono uwagę na te zastosowania programowania logicznego, które rewolucjonizują cykl produkcyjny oprogramowania i w wielu wypadkach pozwalają obejść się całkowicie bez fazy implementacji programu, a nawet jego specyfikacji. Możliwe są też oczywiście inne zastosowania sztucznej inteligencji w inżynierii oprogramowania, na które zwróciłby uwagę projektant oprogramowania, np.: inteligentne narzędzia wspomagające konwencjonalny proces tworzenia oprogramowania, inteligentne sprzężenia (ang. front-ends) z konwencjonalnymi, lecz mało komunikatywnymi programami, bazy wiedzy do wspomagania konwencjonalnego procesu tworzenia oprogramowania, systemy ekspertowe mające wiedzę z dziedziny konwencjonalnej inżynierii oprogramowania. Takie zastosowania sztucznej inteligencji, choć są naturalne, nie powinny jednak odwracać uwagi od możliwości tej dziedziny, zdolnej całkowicie zrewolucjonizować technologię produkcji oprogramowania.

Istnieją też takie zastosowania programowania logicznego w inżynierii oprogramowania, które już teraz mają ogromne znaczenie i perspektywy rozwoju. W szczególności dziedziną, która łączy problematykę sztucznej inteligencji i inżynierii oprogramowania, jest formalne, wspomagane komputerem wyprowadzanie programów ze specyfikacji. Wyprowadzanie formalne jest potrzebne wówczas, gdy wykonywalna analiza systemowa nie funkcjonuje wystarczająco efektywnie, aby sprostać wymaganiom użytkownika. Taka potrzeba nie występowała na ogół w wypadku forma-

lizacji Brytyjskiego Aktu Narodowości, chociaż i tam używano techniki transformacji programów (zastosowanej ręcznie), aby wyeliminować niektóre pętle.

W wielu innych wypadkach, jak na przykład w sortowaniu plików danych, wykonanie analizy danej dziedziny jest niewystarczające. Trzeba zwiększyć jej efektywność przez ograniczenie klasy rozwiązywanych problemów i ograniczenie wiedzy, tak aby była bardziej ukierunkowana na daną klasę. Zmienia to analizę systemu na wykonywalną specyfikację systemu, która w dalszym ciągu jest zapisana w tym samym języku opartym na logice. Teraz jednak specyfikacja wygląda formalnie i ściśle. Wrażenie to jest jednak powierzchowne, ponieważ składniowo nie ma różnicy między językiem specyfikacji formalnej a językiem doświadczalnie przeprowadzonej analizy systemowej.

Jeśli wykonywalna, oparta na logice specyfikacja wciąż nie jest wystarczająco efektywna, to może być dalej przekształcona w bardziej od niej efektywny program. Jeśli trzeba, program może być zapisany w konwencjonalnym języku programowania. Jednak przy odpowiednich zasobach programowych i sprzętowych specyfikacja może być wyrażona w tym samym języku opartym na logice. Użycie tego samego języka do wszystkich faz cyklu produkcyjnego oprogramowania poważnie upraszcza problem zachowania zgodności między różnymi fazami. Ponadto techniki transformacji i wyprowadzania, które gwarantują zachowanie poprawności, mogą być użyte do przejścia od jednej fazy do drugiej. Rozwinęli je głównie badacze języków deklaratywnych, pracujący na pograniczu sztucznej inteligencji i inżynierii oprogramowania.

Takie są zastosowania sztucznej inteligencji w inżynierii oprogramowania. A jakie są zastosowania inżynierii oprogramowania w dziedzinie sztucznej inteligencji? Projektant oprogramowania troszczy się o jego trzy cechy, na które nie zawsze zwraca się dostateczną uwagę w sztucznej inteligencji: poprawność, rozmiary i złożoność oprogramowania.

Jak próbowano przekonać Czytelnika, wiele zastosowań sztucznej inteligencji lepiej jest potraktować jako wykonywalną analizę czy też wykonywalną specyfikację niż jako wykonywanie konwencjonalnych programów. Zastosowania tego rodzaju są równie poprawne jak każda konwencjonalna analiza lub specyfikacja systemu. Jednakże wiele zastosowań sztucznej inteligencji wykracza poza analizę i specyfikację, gdy w grę wchodzą zagadnienia efektywności. W takich wypadkach otrzymane programy wymagają atestacji (ang. validation) i weryfikacji w takim samym stopniu jak każdy program konwencjonalny. Projektanci oprogramowania mają słuszność, krytykując programistów używających takich technik sztucznej inteligencji, którym brakuje podstaw logicznych lub które nie poddają się ścisłej weryfikacji.

Jest to obszar, w którym reprezentacja wiedzy i programowanie z wykorzystaniem logiki mają wyraźną przewagę nad innymi metodami, np.: reprezentacją wiedzy za pomocą ram (ang. frames) lub nad programowaniem ukierunkowanym obiektowo (ang. object-oriented programming). Reprezentacja wiedzy i programy wyrażone w języku logiki są zapisane za pomocą tego samego formalizmu, którego projektant oprogramowania używa do wyrażania specyfikacji formalnych. Korzystanie z tego samego języka, opartego na logice, zarówno do zapisywania programów, jak i specyfikacji, znacznie upraszcza dowodzenie poprawności.

Co można powiedzieć o pozostałych wymienionych cechach: rozmiarach i złożoności oprogramowania? Obawiam się, że niewiele więcej poza powtórzeniem rady DeMarco o używaniu co najwyżej pojedynczej kartki papieru dla pojedynczego diagramu przepływu danych (lub dla równoważnego zbioru reguł, niezależnie od tego czy reprezentują one analizę, specyfikację czy program). Może się jednak zdarzyć, że inne podejścia, np. programowanie ukierunkowane obiektowo mogą wnieść tu coś użytecznego. Jeśli tak, to sądzę, że wówczas będą one wymagały zintegrowania z podejściem opartym na logice, prawdopodobnie zgodnie z sugestiami zawartymi w artykule [5].

SKUTKI ISTOTNE DLA CZŁOWIEKA

Warto teraz zająć się niektórymi skutkami nowej technologii, które mogą okazać się istotne dla człowieka.

Nie wydaje się, aby technika i jej rozwój były same w sobie rozsądnymi celami działalności ludzkiej. Sądzę natomiast, że techniki programowania sztucznej inteligencji oparte na wiedzy mogą przyczynić się do podniesienia przeciętnego poziomu życia. Jednakże, brak świadomości pewnych potencjalnych niebezpieczeństw i niepodejmowanie odpowiednich środków ostrożności, może wywołać spektakularne, niepożądane skutki.

Bardzo łatwo jest pozwolić komputerom na zawładnięcie nami i podejmowanie decyzji, które powinniśmy podejmować sami. Pozwoliliśmy już na to niektórym ludziom, zawodowym doradcom. Lekarza obarcza się odpowiedzialnością za nasze zdrowie, księgowemu pozwala się prowadzić nasze sprawy finansowe. Ekspert może nas onieśmielać mając większą wiedzę i doświadczenie. Jeśli ludzie mogą onieśmielać ludzi, to mogą to czynić i będą to czyniły także komputery, gdy entuzjastyczni zwolennicy techniki będą mogli postawić na swoim. Inżynierowie będą z przyjemnością projektowali komputery, które coraz częściej będą wyręczały człowieka w myśleniu.

Choć można zauważyć rzeczywiste niebezpieczeństwa nowej technologii, widoczne są także duże potencjalne korzyści. Nowa technika obliczeniowa znajduje oczywiste zastosowania w budowie inteligentnych sprzężeń, i to nie tylko do konwencjonalnego oprogramowania, lecz również do różnego rodzaju „nieprzyjaznych" urządzeń — na przykład, mojej własnej kuchenki. Nie cierpię mojej kuchenki, gdyż nie wiem, jak z niej korzystać we właściwy sposób; ona zaś nie wie, jak skorzystać z moich umiejętności przyrządzania potraw. Pośrednictwo czegoś, co bardziej przypominałoby maszynę i wobec tego byłoby bliższe mojej kuchence niż ja sam, co jednak mogłoby rozumieć świat w sposób zbliżony do ludzkiego sposobu rozumienia, a nie tak, jak rozumieją świat dzisiejsze komputery, może uczynić świat mechanizmów bardziej przyjazny i zrozumiały.

Nowe języki oparte na logice umożliwiają zmianę typowego zachowania współczesnych komputerów, charakteryzującego się postawą „wszystko lub nic". Pozwalają one bowiem komputerom wyjaśnić wyprowadzone wnioski, dzięki czemu ułatwiają decydowanie o tym, czy wnioski te są dla nas do przyjęcia. Jedynie wtedy, gdy programy komputerowe są wyrażone w jawnej, deklaratywnej postaci, można stwierdzić, z jakich założeń korzystają oraz można podjąć decyzje co do przyjęcia tych założeń i wynikających z nich konkluzji.

Komputery takie mogą także podnieść poziom wiedzy i umiejętności człowieka. Skomputeryzowane encyklopedie zaczynają już ułatwiać szybki dostęp do całego bogactwa jawnie sformułowanej wiedzy ludzkiej. Dzięki technice wydobywania wiedzy, informacje tkwiące jedynie w podświadomości człowieka mogą być wypowiedziane i wydobyte na światło dzienne. Podobnie jak system ekspertowy może pomóc zrozumieć nieuświadomioną wcześniej wiedzę i przekonania eksperta z dziedziny medycyny, tak technika wydobywania wiedzy może zapewnić lepsze zrozumienie nie tylko ekspertów, lecz także innych ludzi.

W ten sposób można nie tylko pogłębić ludzką wiedzę, lecz również usprawnić ludzkie rozumowanie. Skoro tylko wiedza zostanie jawnie zapisana, będzie można lepiej sobie uświadomić, w co naprawdę się wierzy. Będzie można zrozumieć przekonania innych, określić poszczególne kroki, które wiodą do wyjaśnienia i umotywowania wiedzy i przekonań. Dzięki lepszemu zrozumieniu siebie i innych będzie można myśleć bardziej racjonalnie. Będzie można zrezygnować z naszych przypuszczeń, gdy dowiemy się, co one znaczą. Będzie można chwilowo przyjąć przekonania innych i wyciągając z nich wnioski zobaczyć, dokąd one prowadzą. Sądzę, że ten nowy sposób myślenia przyczyni się do polepszenia świata, w którym żyjemy.

Podsumowując sądzę, że zautomatyzowanie logiki usprawni technikę obliczeniową i stanie się kluczem do rozwoju technik komputerowych nowej generacji. Jest to ponadto ogniwo łączące języki reprezentacji wiedzy w dziedzinie sztucznej inteligencji i języki analizy systemowej, języki specyfikowania programów i języki baz danych w inżynierii oprogramowania.

Jednakże najważniejsze nie są komputery, inżynieria oprogramowania ani sztuczna inteligencja, lecz ludzie. O ile zastosujemy właściwe środki ostrożności, to sądzę, że nowa technologia pozwoli nam stać się bardziej ludzkimi, pomagając zrozumieć lepiej siebie i innych.

LITERATURA

[1] Clark K. L., McCabe F.: Micro-Prolog — Programming in LOGIC. Prentice-Hall, London, 1984
[2] Clocksin W. F., Mellish C. S.: Programming in Prolog. Springer-Verlag, Berlin, 1981
[3] Hammond P.: Representation of DHSS regulations as a logic program. G. Mitra (ed.), Computer Models for Decision Making — Mathematical Programming. Decision Analysis. Expert Systems. North-Holland, Amsterdam, 1983
[4] Hammond P., Sergot M. J.: APES Reference Manual. Logic Based Systems Ltd., Richmond (Surrey), 1984
[5] Hayes P. J.: The Logic of Frames, pp. 46—61; D. Metzing (ed.), Frame Conceptions and Text Uunderstanding. Walter de Gruyter, Berlin, 1979
[6] Hogger C. J.: Introduction to Logic Programming. Academic Press, London, 1984
[7] Kowalski R. A.: Logic for Problem Solving. Elsevier/North-Holland, New York, 1979
[8] Moto-oka T. et al.: Fifth Generation Computer Systems. North-Holland, Amsterdam, 1982
[9] Sergot M. J., Sadri F., Kowalski R. A., Kriwaczek F., Hammond P., Cory H. T.: The British Nationality Act as a logic Program Department of Computing, Imperial College, London, 1984.

MAREK PAWŁOWSKI
Instytut Informatyki
Politechnika Warszawska

Mikroprogramowanie (I)

„Słownik informatyki" (WNT, Warszawa 1981) podaje, że mikroprogramowanie to przygotowanie i realizacja sekwencji rozkazów elementarnych (mikroprogramu) przechowywanych w specjalnej pamięci, odpowiadających operacjom komputera, których wykonanie inicjuje wprowadzenie rozkazu do rejestru rozkazów. Rozkazy elementarne są nazywane mikrorozkazami. Składają się na nie mikrooperacje reprezentujące działanie poszczególnych elementów komputera, takich jak rejestry, sumatory itp. Do zapisu mikroprogramów służą specjalne języki mikroprogramowania.

W początkowym okresie rozwoju komputerów funkcje pamięci mikroprogramu pełniły matryce diodowe (w matrycę taką była między innymi wyposażona ODRA 1204). W miarę postępu w budowie sprzętu komputerowego, poziom mikroprogramowania został przysłonięty przez listę rozkazową komputera, a ta — przez języki wysokiego poziomu.

Pojawienie się uniwersalnych układów mikroprocesorowych dało możliwość użycia ich do budowy większości urządzeń cyfrowych. Jednak realizacja skomplikowanych obliczeń arytmetycznych za pomocą mikrokomputerów 8-bitowych stwarza pewne problemy, których przyczyną jest mała szybkość działania tych układów i uboga w operacje arytmetyczne lista instrukcji. Przykładowo czas wykonania mnożenia liczb reprezentowanych słowami 8-bitowymi dla układów 8080A, Z80, MC6800 wynosi około 200 μs.

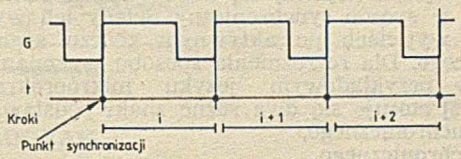
Wyposażenie mikroprocesorów 16-bitowych w instrukcje mnożenia i dzielenia oraz kilkukrotne zwiększenie szybkości działania (względem układów 8-bitowych) zmniejszyło czas mnożenia do kilkudziesięciu mikrosekund dla argumentów 16-bitowych (MC68000 — mnożenie w czasie około 9 μs, 8086 — 27 μs). Jednak w niektórych zastosowaniach to nie wystarcza. Przykładem może być profesjonalna grafika komputerowa, wymagająca w wypadku animacji obrazu wykonania skomplikowanych obliczeń arytmetycznych na liczbach zmiennoprzecinkowych w bardzo krótkim czasie, np. kilkaset nanosekund. Do tego celu można jedynie wykorzystać układy mikroprogramowe, na przykład serii Am29500.

Wydawać by się mogło, że mikroprogramowanie to tylko historia, ale — jak wykazuje praktyka — można za jego pomocą rozwiązać zadania, dla których inne metody okazały się nieskuteczne. Prostota projektowania układów mikroprogramowych sprawia, że technika ta jest wykorzystywana między innymi do budowy nowych układów scalonych o dużym stopniu scalenia. Tak też powstały układy 8086 i MC68000, których działanie jest określane przez mikroprogramowany układ sterowania.

Mgr inż. Marek PAWŁOWSKI ukończył w 1977 r. studia na Wydziale Elektroniki Politechniki Warszawskiej. Od 1977 r. pracuje w Instytucie Informatyki Politechniki Warszawskiej. Jest konstruktorem systemów mikroprocesorowych i urządzeń mikroprogramowanych, a w szczególności licznej serii programatorów pamięci stałych. Na Wydziale Elektroniki Politechniki Warszawskiej prowadzi m.in. wykład poświęcony mikroprogramowaniu.

MIKROPROGRAMOWANE ZESPOŁY FUNKCJONALNE

Mikroprogramowane zespoły funkcjonalne (MZF) są synchronicznymi automatami sekwencyjnymi, których działanie zależy od zawartości pamięci mikroprogramu. MZF są synchronizowane jednym lub wieloma sygnałami zegarowymi. Dla uproszczenia w artykule tym przyjęto, że MZF jest synchronizowany tylko jednym sygnałem zegara G.



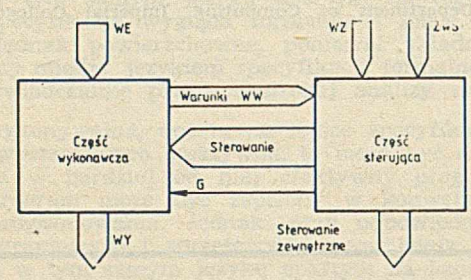
Rys. 1. Podział czasu na kroki przetwarzania. Przyjęto, że aktywnym zboczem sygnału G jest zbocze narastające

Sygnał ten dzieli oś czasu rzeczywistego na przedziały, zwane krokami przetwarzania. Krokiem przetwarzania (rys. 1) jest lewostronnie otwarty przedział czasu między dwoma kolejnymi punktami osi czasu, wyznaczonymi przez aktywne zbocze sygnału synchronizującego. Algorytm przetwarzania informacji w MZF można więc przedstawić jako listy operacji wykonywanych w MZF w kolejnych krokach przetwarzania. Listy operacji opisują działanie poszczególnych elementów składowych MZF. Można zatem podzielić MZF na mniejsze elementy, cyfrowe bloki funkcjonalne, które, jak się później okaże, daje się projektować niezależnie od siebie. Zamiast więc projektować jeden duży układ składający się z kilkudziesięciu lub nawet kilkuset układów scalonych, można po prostu zbudować go z bloków funkcjonalnych.

W każdym mikroprogramowanym zespole funkcjonalnym wyróżnia się dwie części składowe (rys. 2): część wykonawczą i część sterującą.

Część wykonawcza realizuje przetwarzanie informacji wejściowej WE w informację wyjściową WY w sposób

wyznaczony przez sterowanie z części sterującej. Podczas przetwarzania danych mogą wystąpić pewne sytuacje decydujące o dalszym wykonaniu algorytmu działania MZF, np. pojawienie się nadmiaru lub zerowego wyniku operacji w ALU; są one sygnalizowane części sterującej przez stan linii warunków WW. Zadaniem części sterującej jest wytworzenie odpowiedniej sekwencji sygnałów sterujących częścią wykonawczą na podstawie stanu warunków wewnętrznych WW, warunków zewnętrznych WZ i zewnętrznych warunków synchronizacji ZWS mających wpływ na sygnał synchronizujący G.



Rys. 2. Podział MZF na część wykonawczą i sterującą

Od układu sterującego jest uzależnione działanie części wykonawczej, którą można traktować jako cyfrowy blok funkcjonalny, ale z praktycznego punktu widzenia lepiej jest podzielić ją na mniejsze bloki funkcjonalne. Bloki te mogą być sterowane niezależnie od siebie (mikroprogramowanie poziome), albo te same linie sterujące mogą wpływać na działanie wszystkich bloków funkcjonalnych części wykonawczej (mikroprogramowanie pionowe). W pierwszym wypadku jest konieczna duża liczba linii sterujących, ale bardzo łatwo można modyfikować algorytm działania MZF. W drugim wypadku otrzymuje się małą liczbę linii sterujących częścią wykonawczą, często bez możliwości wprowadzania poprawek do sposobu sterowania MZF.

Pierwszym etapem projektowania części wykonawczej MZF jest podział jej zadań między składające się na nią cyfrowe bloki funkcjonalne.

CYFROWE BLOKI FUNKCJONALNE

Przystępując do projektowania cyfrowego bloku funkcjonalnego należy określić listę mikrooperacji realizowaną przez ten blok w kolejnych krokach przetwarzania. Najlepszym sposobem opisu projektowanego bloku jest przedstawienie go w postaci tabeli mikrooperacji. Każdy z wierszy tabeli powinien zawierać zestaw mikrooperacji wykonywanych przez blok funkcjonalny w jednym kroku. Mikrooperacje mogą być wykonywane w sposób asynchroniczny, wtedy ich wyniki pojawią się na wyjściach bloku natychmiast; lub w sposób synchroniczny, wtedy ich wyniki wystąpią na wyjściach po aktywnym zboczu sygnału synchronizującego. Dla rozróżnienia sposobu wykonania mikrooperacji w przykładowym języku mikroprokramowania MIDDLE [2] stosuje się dwa różne znaki podstawienia:
⟨= — asynchronicznego,
:= — synchronicznego.

Zapis mikrooperacji powinien spełniać dwa warunki:

- po pierwsze powinien możliwie jak najdokładniej odzwierciedlać działanie układu, tak aby mógł zastąpić jego schemat logiczny;
- po drugie, użycie danej mikrooperacji w opisie działania MZF powinno z jednej strony dokładnie wyznaczać sterowanie blokiem, który ją wykonuje, a z drugiej — umożliwiać zrozumienie działania algorytmu. Wynika stąd, że do opisu działania bloku nie można użyć operatorów arytmetycznych. Konieczne jest zdefiniowanie odpowiedniego zestawu funkcji z podaniem ich dziedziny i przeciwdziedziny.

Przykładem definicji funkcji, która będzie nazywać się mikrooperacją, jest opis działania sumatora dwójkowego:

- mikrooperacja generowania słowa przeniesień przy dodawaniu arytmetycznym słowa A do słowa B z uwzględnieniem przeniesienia początkowego p:

— deklaracja dziedziny i przeciwdziedziny (word[—1:—m] oznacza słowo dwójkowe n+m—pozycyjne)

CAR(A,B:word[n—1:—m],p:binarny):word[n:—m]

— opis wykonania mikrooperacji

$CAR(A,B,p)[-m] = p$

$CAR(A,B,p)[i+1] = A[i] \wedge B[i] \vee (A[i] \vee B[i]) \wedge CAR(A,B,p)[i]$

dla $i = -m, \quad -m+1, \cdots n-2, \quad n-1$

- mikrooperacja dodawania słowa A do słowa B z uwzględnieniem bitu p:

AD(A,B:word[n—1:—m],p:binary):word[n—1:—m]

$AD(A,B,p)[i] = A[i] \oplus B[i] \oplus CAR(A,B,p)[i]$

dla $i = -m, ..., n-1$, gdzie $\oplus$ oznacza operator EXOR.

Należy zwrócić uwagę, że obie mikrooperacje są zdefiniowane jako działanie na słowach dwójkowych, a nie na liczbach, dlatego też są one niezależne, podobnie jak sumator dwójkowy, od kodu, w jakim liczby są reprezentowane w układach cyfrowych.

Dopiero interpretacja liczbowa argumentów i wyników działania tych mikrooperacji w odpowiednim kodzie da w rezultacie operację arytmetyczną na liczbach. Do wyznaczania interpretacji lizbowej wyników może być użyta zależność arytmetyczna między stanami wejść i wyjść i-tego bloku sumatora dwójkowego:

$$A[i] + B[i] + CAR(A,B,p)[i] = AD(A,B,p)[i] + 2 * CAR(A,B,p)[i+1].$$

Korzystając z definicji obu mikrooperacji i zależności $B(X) = \sum_{i=-m}^{n-1} x[i] * 2^i$ dla kodu naturalnego binarnego (gdzie B(X) jest wartością liczby reprezentowanej przez słowo X: word[n—1:—m] w kodzie NB), można wyprowadzić następującą równość:

$$B(AD(A,B,p)) = B(A) + B(B) + p * 2^{-m} - CAR(A,B,p)[n].$$

Wynik wykonania dodawania słów A i B reprezentujących liczby w kodzie NB jest słowem reprezentującym ich sumę, o ile przeniesienie wyjściowe z sumatora jest równe 0 CAR(A,B,p)[n]=1 jest więc warunkiem nadmiaru przy sumowaniu liczb w kodzie NB.

Za pomocą mikrooperacji AD i CAR można więc zdefiniować zestaw mikrooperacji arytmetycznych, na przykład:

SUB(A,B) = AD(A,$\overline{B}$,1) — odejmowanie słowa B od słowa A,

DSB(A,B) = AD(A,$\overline{B}$,0) — odejmowanie słowa B od słowa A z dekrementem,

INC(A)=AD(A,[0],1) — zwiększenie o 1 słowa A (zapis [0] oznacza słowo word[n—1:—m] składające się z samych zer),

DEC(A)=AD(A,[1],0) — zmniejszenie o 1 słowa A.

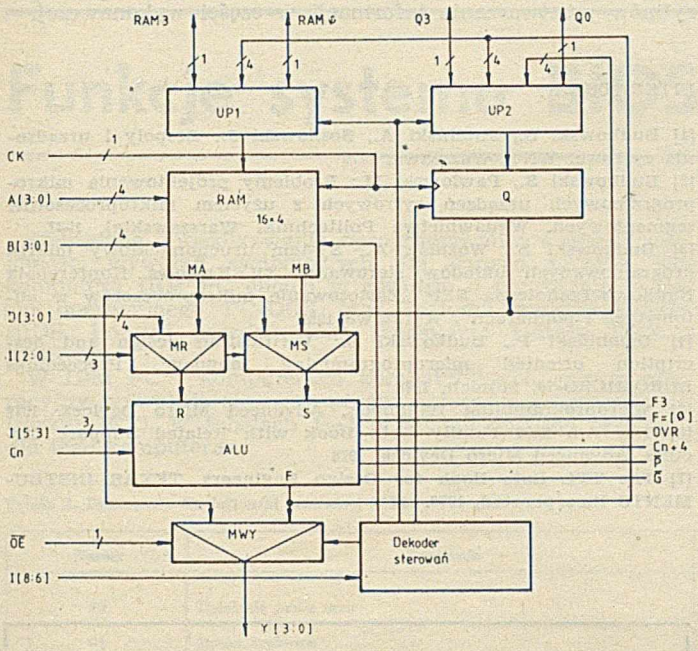
Poniżej przedstawiono przykładową tabelę mikrooperacji opisującą działanie licznika rewersyjnego SN74191 [6] (tab. 1). Cyfrowym blokiem funkcjonalnym może być rejestr, połączenie równoległe układów sumatora dwójkowego itp. Również mikroprocesory segmentowe można traktować jako wykonane w jednym układzie scalonym połączenie kilku bloków funkcjonalnych przetwarzających informację 2-, 4- lub 8-bitową.

Tabela 1. Mikrooperacje układu SN74191 (Q reprezentuje wyjścia licznika, a WE—słowo do niego zapisywane)

CK	Stan wejść sterujących X	D/U	$\overline{LD}$	Mikrooperacje
X	X	0	0	Q ⇐ WE, MM ⇐ (Q = [1])
X	X	1	0	Q ⇐ WE, MM ⇐ (Q = [0])
↑	1	0	1	Q := Q, MM ⇐ (Q = [1])
↑	1	1	1	Q := Q, MM ⇐ (Q = [0])
↑	0	0	1	Q := INC(Q), MM ⇐ (Q = [1])
↑	0	1	1	Q := DEC (Q), MM ⇐ (Q = [0])

MIKROPROCESOR SEGMENTOWY Am2901

Jednym z najbar ej rozpowszechnionych mikroprocesorów segmentowych jest układ Am2901 produkowany od 1976 roku przez firmę Advanced Micro Devices (rys. 3).



Rys. 3. Schemat blokowy układu Am2901

Pojedynczy układ Am2901 umożliwia przetwarzanie danych 4-bitowych. Równoległe połączenie n tych układów pozwala budować procesory działające na n×4-bitowym słowie dwójkowym. Jednym z podstawowych bloków funkcjonalnych układu Am2901 jest dwuadresowa pamięć notatnikowa RAM o organizacji 16 słów 4-bitowych. W stanie wysokim sygnału zegara CK następuje odczytanie komórek pamięci o adresach podanych na wejścia A i B układu. Jeśli stan wejść A i B jest taki sam, to jest odczytywana ta sama komórka RAM. Pod wpływem opadającego zbocza sygnału CK następuje podtrzymanie odczytanej informacji w rejestrach zatrzaskowych umieszczonych na wyjściach MA i MB pamięci notatnikowej. W stanie niskim zegara CK do komórki pamięci RAM wskazanej przez adres B może zostać zapisana dana, otrzymana z wyjścia układu UP1. O zapisie do RAM i działaniu układu UP1 decyduje sterowanie I[8:6] (tab. 4). Sposób dostępu do RAM umożliwia wykonanie w jednym cyklu zegara CK operacji odczytania argumentów z pamięci notatnikowej, przetworzenia ich w jednostce arytmetyczno-logicznej ALU, przesunięcia wyniku w układzie UP1 i zapisania go do wskazanej komórki tej samej pamięci.

Zadaniem pozostałych bloków układu Am2901 jest:

MR, MS — wybór argumentów operacji w ALU (tab. 2);

ALU — realizacja dwuargumentowych operacji arytmetyczno-logicznych (tab. 3) z podaniem dodatkowych informacji o wyniku wykonanej operacji:
— F=[0] — wynik jest równy zero (wyjście z otwartym kolektorem),
— F3 — znak wyniku,
— OVR — nadmiar przy sumowaniu słów reprezentujących liczby w kodzie U2 (ma znaczenie tylko dla najstarszego segmentu),

$\overline{P}, \overline{G}$ — warunki propagacji i generacji przeniesień;

MWY —wybór danej wyprowadzonej na wyjście Y układu;

UP1 — realizacja operacji przesuwania w lewo lub w prawo informacji zapisywanej do RAM; linie RAM3 i RAM0 są dwukierunkowe, co umożliwia odczytanie informacji traconej w wyniku przesunięcia lub wpisania bitu na zwalnianą pozycję;

UP2 — realizacja operacji przesunięcia w lewo lub w prawo zawartości rejestru Q; linie Q3 i Q0 mają podobne znaczenie jak RAM3 i RAM0;

Q — rejestr pomocniczy.

Tabela 2. Sterowanie multiplekserami MR i MS

I[2:0]	Mikrooperacje
0	R ⇐ RAM[A], S ⇐ Q
1	R ⇐ RAM[A], S ⇐ RAM[B]
2	R ⇐ [0], S ⇐ Q
3	R ⇐ [0], S ⇐ RAM[B]
4	R ⇐ [0], S ⇐ RAM[A]
5	R ⇐ D, S ⇐ RAM[A]
6	R ⇐ D, S ⇐ Q
7	R ⇐ D, S ⇐ [0]

Działanie układu Am2901 opisują tabele mikrooperacji (tab. 2, 3, 4). Dokładny opis działania układu Am2901 oraz jego rozbudowanej wersji, układu Am2903 można znaleźć w [2] i [6].

Tabela 3. Sterowanie blokiem ALU

I[5:3]	Mikrooperacje
0	F ⇐ AD(R,S,Cn)
1	F ⇐ AD($\overline{R}$,S,$\overline{Cn}$) (*)
2	F ⇐ AD(R,$\overline{S}$,$\overline{Cn}$) (*)
3	F ⇐ OR(R,S)
4	F ⇐ AND(R,S)
5	F ⇐ AND($\overline{R}$,S)
6	F ⇐ EXOR(R,S)
7	F ⇐ EXOR($\overline{R}$,S)

(*) — operacje odejmowania

MIKROPROGRAM

Działanie cyfrowych bloków funkcjonalnych opisuje się za pomocą powyższych tabel mikrooperacji. Do zapisu algorytmu działania mikroprogramowanych zespołów funkcjonalnych służą specjalne języki mikroprogramowania. Jednym z nich jest język MIDDLE (Microprogram Design and

Tabela 4. Sterowanie pozostałymi blokami układu Am2901 (IN3, IN0, IQ3, IQ0 — sygnały binarne wprowadzane podczas przesunięcia na zwalniane pozycje słowa zapisywanego do RAM lub Q)

I[8:6]	Mikrooperacje
0	RAM[B] := RAM[B], Q := F, Y⇐F
1	RAM[B] := RAM[B], Q := Q, Y⇐F
2	RAM[B] := F, Q := Q, Y⇐RAM[A]
3	RAM[B] := F, Q := Q, Y⇐F
4	RAM[B] := SHR(IN3, F), Q := SHR(IQ3, Q), Y⇐F, RAM0⇐F[0], Q0⇐Q[0]
5	RAM[B] := SHR(IN3, F), Q := Q, Y⇐F, RAM0⇐F[0], Q0⇐Q[0]
6	RAM[B] := SHL(F, IN0), Q := SHL(IQ0, Q), Y⇐F, RAM3⇐F[3], Q3⇐Q[3]
7	RAM[B] := SHL(F, IN0), Q := Q, Y⇐F, RAM3⇐F[3], Q3⇐Q[3]

Description Language [4]). Język ten charakteryzuje możliwość zapisu algorytmu działania MZF na różnym poziomie, od algorytmu na liczbach rzeczywistych do opisu na poziomie przesłań międzyrejestrowych.

Jak w każdym języku mikroprogramowania jednemu cyklowi zegara odpowiada jeden mikrorozkaz opisujący działanie wszystkich bloków funkcjonalnych części wykonawczej MZF, za pomocą mikrooperacji występujących w tabelach mikrooperacji bloków składowych. W mikrorozkazie można wyróżnić trzy elementy składowe:

- etykietę mikrorozkazu,
- listę mikrooperacji wykonawczych, tzn. realizowanych przez bloki funkcjonalne części wykonawczej,
- mikrooperację rozejścia wskazującą etykietę następnego mikrorozkazu oraz warunek przejścia do niego.

Lista mikrorozkazów tworzy część wykonawczą mikroprogramu. Jest ona poprzedzona częścią deklaracyjną, w której konstruktor MZF podaje typ zmiennych reprezentujących elementy sprzętowe zespołu oraz definiuje funkcje umożliwiające mu zapis działania bloków cyfrowych MZF. Przykładem definicji funkcji są przedstawione wcześniej definicje mikrooperacji AD(A,B,p) i CAR(A,B,p). Zmienne występujące w mikroprogramie mogą być między innymi typu:

binary — typ binarny,
array [n—1:—m] of binary — typ słowowy
(zapis skrócony word [n—1:—m]),
integer — liczby całkowite,
real — liczby rzeczywiste.

Część deklaracyjna jest ujęta w nawiasy lcd ... dcl. Pełną definicję języka MIDDLE podano w [2] i [4].

Przykładem zapisu mikroprogramu w języku MIDDLE może być mikroprogram mnożenia metodą podstawową liczb ułamkowych reprezentowanych w kodzie NB słowami MN, MK: word [0:—7], podany w pozycji [5]. Część wykonawcza MZF jest zbudowana z dwóch układów Am2901. W RAM[0] będzie przechowywana mnożna, w Q — mnożnik, a w RAM[1] iloczyn częściowy. Licznik pozycji mnożnika zrealizowany w układzie sterującym jest reprezentowany przez zmienną L. Dla uproszczenia przykładu pominięto część deklaracyjną mikroprogramu.

```
M1: D ⇐ MN, F ⇐ AD(D,[0],0),   RAM[0] := F
                    exit  Z = 0. M1,   Z = 1. M2;
M2: D ⇐ MN,   F ⇐ AD(D,[0],0),   Q := F
                    exit Z = 1, M2.   Z = 0. M3;
M3: F ⇐ AND([0], RAM[1]),   RAM [1] := F,
    N0:   L := 7   exit   M4;
M4: F ⇐ OR([0], RAM[1]),   RAM[1] := SHR(N,F),
    Q := SHR(F[—15], Q),   Q0 ⇐ Q[—15]
                    exit Q0 = 0. M5,   Q0 = 1. M6;
M5: F ⇐ AD([0], RAM[1],0),   RAM[1] := F,
    N := CAR([0], RAM[1], 0)[1],   L := DEC(L)
                    exit L ≠ [0]. M4, L = [0]. M7;
M6: F ⇐ AD(RAM[0], RAM[1],0),   RAM[1] := F,
    N := CAR(RAM[0], RAM[1],0)[1],   L := DEC(L)
                    exit L ≠ [0]. M4, L = [0]. M7;
M7: F ⇐ OR([0], RAM[1]),   RAM[1] := F,   Y ⇐ F
                    exit O = 0. M7,   O = 1. M1;
```

W mikrorozkazie M1 występuje mikrooperacja rozejścia „exit Z = 0.M1, z = 1.M2;". Gdy zewnętrzny sygnał sterujący Z będzie równy 0, wówczas w kolejnym takcie zegara MZF będzie ponownie wykonywał mikrorozkaz M1. Jeżeli Z = 1, to w kolejnym cyklu zegara zostanie wykonany mikrorozkaz o etykiecie M2. Zmienna Z potwierdza ważność danych wprowadzonych do MZF, a zmienna O = 1 sygnalizuje odczytanie wyniku mnożenia. Po wykonaniu mikroprogramu stan zmiennej N wskazuje, czy wystąpił nadmiar operacji mnożenia.

Z powyższego przykładu widać, że w mikroprogramie wyraźnie występuje podział na opis działania części wykonawczej i sterującej. Działanie części wykonawczej MZF opisują mikrooperacje wykonawcze, a działanie części sterującej „wyrażają" mikrooperacje rozejść. Jest to wynikiem wyraźnego ukierunkowania języka MIDDLE na zapis algorytmów przetwarzania informacji w części wykonawczej.

LITERATURA

[1] Budkowski S., Papliński A., Sosnowski J.: Zespoły i urządzenia cyfrowe. WNT, Warszawa, 1978
[2] Budkowski S., Pawłowski M.: Problemy projektowania mikroprogramowych urządzeń cyfrowych z użyciem mikroprocesorów segmentowych. Wydawnictwa Politechniki Warszawskiej, 1987
[3] Budkowski S., Woźniak A.: System uruchomieniowy mikroprogramowanych układów sterowania. IV Krajowa Konferencja Naukowo-Techniczna SEP „Zastosowanie mikroprocesorów w automatyce i pomiarach", Warszawa 1984
[4] Dembiński P., Budkowski S.: Verification design and description oriented microprogramming language. Proceedings EUROMICRO-78, Münich, 1978
[5] Microprogramming Handbook. Advanced Micro Devices, 1976
[6] The Am 2900 Family Data Book with Related Support Circuits. Advanced Micro Devices, 1978
[7] The TTL Data Book for Design Engineers. TEXAS INSTRUMENTS Incorporated, 1976.

Stała sprzedaż egzemplarzy
MIKROKLANU
prowadzona jest
w Klubie Prasy Technicznej
w Warszawie
ul. Mazowiecka 12, tel. 27-43-65

Ceny ogłoszeń

Od 1 stycznia br. obowiązują następujące ceny ogłoszeń publikowanych na naszych łamach:
ogłoszenia duże (zależnie od objętości):
cała strona — 35 tys. zł, 3/4 — 30 tys., 1/2 — 25 tys., 1/4 — 20 tys., 1/8 — 15 tys.
ogłoszenia drobne (zależnie od liczby słów)
jedno słowo — 30 zł

Dodatki do ceny podstawowej:
— za dodatkowy kolor (na okładce) +30%
— za zamieszczenie ogłoszenia na czwartej stronie okładki +100%
— za zamieszczenie ogłoszenia na trzeciej stronie okładki +50%

Zniżki:
— za ogłoszenie 3—5-krotne —5%
— za ogłoszenia 6—10-krotne —10%
— za ogłoszenia 11-krotne i powyżej —20%
— za artykuły reklamowe i wkładki wykonane przez zleceniodawcę —40%
— za bloki i biuletyny wykonane przez zleceniodawcę — maks. —60%

W przypadku dostarczenia przez zleceniodawcę materiału ilustracyjnego nie odpowiadającego warunkom technicznym druku lub tekstu wymagającego redakcyjnego opracowania, do powyższych cen doliczane będą koszty odpowiednich usług fotograficznych, graficznych lub przygotowania tekstów.

ANDRZEJ RAŻNOWIECKI
ANNA SYFERT
Warszawa

Funkcje systemu BIOS dla PC/XT

Efektywne opracowywanie oprogramowania wymaga znajomości wywołań systemowych dla danego sprzętu. Dla komputera IBM PC chodzi w zasadzie o wywołania systemu operacyjnego PC-DOS oraz systemu wejścia-wyjścia BIOS. Poniżej przedstawiono wywołania łączące DOS z BIOS-em.

W IBM PC i komputerach kompatybilnych system operacyjny wywołuje się przy użyciu przerwań programowych (rozkaz INT). W tabelach 1—4 przedstawiono 256 przerwań dla tego komputera.

Tabela 1. Przerwania dla jednostki centralnej 8088

Numer	Znaczenie
00	Dzielenie przez zero
01	Praca krokowa
02	Przerwanie niemaskowalne NMI
03	Punkt wstrzymania (ang. breakpoint)
04	Nadmiar
05	Drukowanie zawartości ekranu monitora
06	Nie dołączone
07	Nie dołączone

Tabela 2. Przerwania sprzętowe dla IBM PC (układ 8259)

Numer	Znaczenie
08	Czasomierz
09	Klawiatura
0A	Nie dołączone
0B	Kanał szeregowy COM2
0C	Kanał szeregowy COM1
0D	Sterownik dysku stałego
0E	Sterownik dysków elastycznych
0F	Drukarka

Pierwsze 16 wektorów przerwań odpowiada przerwaniom sprzętowym, pozostałe są przewidziane dla przerwań programowych. W zasadzie chodzi tu o funkcje PC-DOS oraz BIOS. Aby PC-DOS mógł wykonywać własne funkcje, BIOS udostępnia mu podstawowe funkcje sprzętowe. Przykładowo, kompleksowy system zarządzania plikami systemu PC-DOS bazuje na podstawowej funkcji systemu BIOS — odczyt i zapis ścieżki lub sektora. Zatem BIOS oferuje podobne funkcje co PC-DOS, lecz bardziej elementarne, np.: wprowadzanie z konsoli, wyprowadzanie na konsolę, obsługa pamięci masowych oraz urządzeń peryferyjnych. Oprócz tego BIOS dysponuje funkcjami graficznymi, funkcjami do inicjalizacji kanałów szeregowych itd., czyli funkcjami, które nie mają odpowiedników w systemie MS-DOS — poprzedniku PC-DOS-a. Z tego powodu wiele programów dla IBM PC nie działa na tzw. komputerach kompatybilnych, ponieważ wykorzystują one bezpośrednio przerwania BIOS-a. Przy dokładnej znajomości funkcji BIOS-a dla IBM PC możliwa jest emulacja tych funkcji na innych komputerach.

Poniżej opisano poszczególne funkcje BIOS-a w formie tabelarycznej. Numery wywołań INT podano w reprezentacji szesnastkowej, zaś parametry wejściowe (np. zawartość rejestru AH) w postaci dziesiętnej.

Tabela 3. Przerwania systemu BIOS

Numer	Znaczenie
10	Ekran monitora
11	Test sprzętowy
12	Wyznaczenie wielkości pamięci
13	Dysk elastyczny
14	Interface szeregowy
15	Magnetofon kasetowy
16	Klawiatura
17	Drukarka
18	ROM-Basic (tylko dla IBM PC)
19	Program samoładujący
1A	Zegar dzienny
1B	Klawisz BREAK
1C	Czasomierz
1D	Inicjalizacja ekranu monitora
1E	Adres parametrów dyskietki
1F	Adres tablicy alfabetu znaków graficznych

Tabela 4. Przerwania systemu operacyjnego PC-DOS i pozostałe

Numer	Znaczenie
20	Normalne zakończenie programu
21	Wywołania funkcji DOS
22	Nowy adres zakończenia programu
23	Obsługa klawisza BREAK
24	Obsługa błędów krytycznych
25	Odczyt fizyczny dysku
26	Zapis fizyczny dysku
27	Zakończenie programu i pozostawienie go rezydentnym
28-3F	Zarezerwowane dla PC-DOS
40-5F	Zarazerwowane dla rozszerzeń
60-7F	Przerwania programowe użytkownika
80-85	Zarezerwowane dla Basica
86-F0	Używane przez interpreter Basica
F1-FF	Nieużywane

INT 10 — funkcje ekranowe

Zawartości rejestrów CS, SS, DS, ES, BX, CX, DX pozostają niezmienione, a zawartości pozostałych rejestrów ulegają zmianie.

AH = 0 — wybór trybu pracy (ekran monitora zostaje oczyszczony)
AL = 0 — tekst o rozdzielczości 40×25 znaków, obraz monochromatyczny

AL = 1 — tekst o rozdzielczości 40×25 znaków, obraz kolorowy
AL = 2 — tekst o rozdzielczości 80×25 znaków, obraz monochromatyczny
AL = 3 — tekst o rozdzielczości 80×25 znaków, obraz kolorowy
AL = 4 — grafika o rozdzielczości 320×200 punktów, obraz kolorowy
AL = 5 — grafika o rozdzielczości 320×200 punktów, obraz monochromatyczny
AL = 6 — grafika o rozdzielczości 640 × 200 punktów, obraz monochromatyczny
AL = 7 — obraz monochromatyczny o rozdzielczości 80×25 znaków
AH = 1 — definicja wielkości kursora
CH — bity 0—4 definiują wiersz początkowy kursora
CL — bity 0—4 definiują wiersz końcowy kursora
Przykładowo, dla karty graficznej można zdefiniować kursor jako podkreślenie, CH = 6, CL = 7, lub jako blok — CH = 0, CL = 7.
AH = 2 pozycjonowanie kursora (punkt o współrzędnych (0, 0) jest w lewym górnym rogu)
DH — numer wiersza
DL — numer kolumny
BH — strona o wielkości ekranu monitora (dla grafiki zawsze 0)
AH = 3 — odczyt położenia kursora
BH — numer strony o wielkości ekranu monitora (parametr wejściowy, dla grafiki zawsze 0)
DX — numer wiersza i kolumny (jak dla AH = 2)
CX — pole kursora (jak dla AH = 1)
AH = 4 — odczyt położenia pióra świetlnego
AH = 0 — pióro świetlne nie jest aktywne
AH = 1 — pióro świetlne jest aktywne
DX — numer wiersza i kolumny
CH — numer wiersza graficznego (0—199)
BX — numer kolumny graficznej (0—319/0—639)
AH = 5 — wybór strony o wielkości ekranu monitora (tylko dla trybów tekstowych)
AL — numer strony
AH = 6 — przesuwanie do góry okna na aktualnej stronie
AL — liczba wierszy
CX — numer wiersza i kolumny dla okna w lewym górnym rogu
DX — numer wiersza i kolumny dla okna w prawym dolnym rogu
BH — atrybut dosuwanych pustych linii (jak dla AH = 8)
AH = 7 — przesuwanie do dołu okna na aktualnej stronie
AL — liczba wierszy
CX — numer wiersza i kolumny dla okna w lewym górnym rogu
DX — numer wiersza i kolumny dla okna w prawym dolnym rogu
BH — atrybut dosuwanych pustych linii (jak dla AH = 8)
AH = 8 — odczyt znaku i atrybutu na pozycji kursora

Tabela 5. Kody kolorów dla karty grafiki kolorowej

Intensywność	Sygnał R	Sygnał G	Sygnał B	Kolor
0	0	0	0	Czarny
0	0	0	1	Niebieski
0	0	1	0	Zielony
0	0	1	1	Zielono-niebieski
0	1	0	0	Czerwony
0	1	0	1	Fioletowy
0	1	1	0	Brązowy
0	1	1	1	Biały
1	0	0	0	Szary
1	0	0	1	Jasnoniebieski
1	0	1	0	Jasnozielony
1	0	1	1	Jasnozielono-niebieski
1	1	0	0	Jasnoczerwony
1	1	0	1	Jasnofioletowy
1	1	1	0	Żółty
1	1	1	1	Jasnobiały

BH — numer strony o wielkości ekranu monitora (parametr wejściowy, dla grafiki zawsze 0)
AL — odczytany znak
AH — odczytany atrybut (tylko dla trybów tekstowych)
Atrybuty dla karty graficznej (p. tabela 5) są następujące:
bity 2, 1, 0 — kolor znaków,
bit 3 — jasność,
bity 6, 5, 4 — kolor tła,
bit 7 — migotanie.
AH = 9 — zapis znaku i atrybutu na pozycji kursora
BH — numer strony o wielkości ekranu monitora (tylko dla trybów tekstowych)
CX — liczba znaków
AL — zapisywany znak
BL — zapisywany atrybut (jak dla AH = 8)
AH = 10 — zapis samego znaku na pozycji kursora (jak dla AH = 9)
AH = 11 — ustawienie palety kolorów dla grafiki
BH — numer palety kolorów do ustawienia
BL — kolor z danej palety kolorów
AH = 12 — ustawianie punktu graficznego
DX — numer wiersza
CX — numer kolumny
AL — bity 0—6 oznaczają kolor, ustawiony bit 7 oznacza, że kolor jest połączony z aktualnym kolorem na danej pozycji ekranu monitora w relacji XOR.
AH = 13 — odczyt koloru punktu graficznego
DX — numer wiersza
CX — numer kolumny
AL — kolor wybranego punktu
AH = 14 — wprowadzenie znaku i przesunięcie kursora o jedną pozycję
AL — znak
BL — kolor znaku
AH = 15 — odczytanie aktualnego stanu ekranu monitora
AL — tryb ekranowy monitora (jak dla AH = 0)
AH — liczba kolumn na ekranie monitora (do 40 lub 80)
BH — aktualny numer strony
INT 11 — test sprzętowy

W czasie tego przerwania następuje odczytanie stanu opcji zespołów sprzętowych zainstalowanych w chwili ładowania systemu operacyjnego:
port 060H — przełącznik do określenia konfiguracji umieszczony na płytce głównej,

port 3FAH — rejestr identyfikacji przerwania układu UART 8250,
port 378H — port drukarki (zapis i odczyt).
Poszczególne bity rejestru AX mają następujące znaczenie:
bit 0 — obecność dysków,
bit 1 — nieużywany,
bity 2, 3 — liczba zespołów pamięci na płytce głównej (1—4),
bity 4, 5 — standardowy tryb pracy ekranu monitora (00 — błędny, 01 — karta kolorowa, obraz monochromatyczny o rozdzielczości 40×25 znaków, 10 — karta kolorowa, obraz monochromatyczny o rozdzielczości 80×25 znaków, 11 — karta monochromatyczna, obraz o rozdzielczości 80×25 znaków),
bity 6, 7 — liczba dysków (jeśli bit 0 jest ustawiony),
bit 8 — nieużywany,
bity 9—11 — liczba interface'ów RS-232,
bit 12 — obecność adaptera do gier,
bit 13 — nieużywany,
bity 14, 15 — liczba interface'ów do drukarki.
INT 12 — wyznaczanie wielkości pamięci

Aktualna wielkość pamięci w chwili ładowania systemu operacyjnego jest określona przez odczyt prrzełącznika konfiguracji:
port 60H, bity 2, 3 — liczba zespołów pamięci na płytce głównej (1—4),
port 62H, bity 0—3 — liczba 32-kilobajtowych zespołów pamięci na płytkach dodatkowych.
W czasie tego przerwania do rejestru AH zostaje wpisana liczba oznaczająca wielkość pamięci w blokach 1-kilobajtowych.

INT 13 — dysk elastyczny
AH = 0 — inicjowanie pracy systemu dyskowego
AH = 1 — przeniesienie stanu ostatniej operacji dyskowej do rejestru AL
AH = 2 — odczyt sektora
AH = 3 — zapis sektora
AH = 4 — odczyt sektora z weryfikacją
AH = 5 — formatowanie ścieżki

Poszczególne parametry dla funkcji 2—5 są następujące:
DL — numer dysku (0—3),
DH — numer głowicy (0—1),
CH — numer ścieżki (0—39),
CL — numer sektora (1—8) (dla funkcji 2—4),
AL — maksymalna liczba sektorów (1—8),
ES:BX — adres bufora (dla funkcji 2, 3 i 5).
Jeżeli podczas wykonywania operacji wystąpi błąd, to ustawiane jest przeniesienie (CARRY = 1), a kod błędu jest zapisywany do rejestru AH:
00H — operacja została wykonana poprawnie,
01H — rozkaz do sterownika jest błędny,
02H — znacznik adresu nie został znaleziony,
03H — błąd zapisu (dyskietka ma ochronę zapisu),
04H — sektor nie został znaleziony,
08H — błąd DMA,
09H — obszar DMA przekracza granicę 64 KB,
10H — błąd CRC,
20H — błąd sterownika dysku 765,
40H — błąd szukania,
80H — błąd przeterminowania (ang. time-out).
Do rejestru AL wpisywana jest liczba odczytanych sektorów (dla funkcji 2—4).
INT 14 — interface szeregowy
AH = 0 — inicjalizacja portu według słowa zapisanego w rejestrze AL:
bity 5—7 — szybkość transmisji (110, 150, 300, 600, 1200, 2400, 4800 i 9600 bodów),
bity 3—4 — parzystość (3), nieparzystość (2) lub brak parzystości (0—1),
bit 2 — liczba bitów stopu (1 lub 2),
bity 0—1 — długość słowa (2 oznacza 7 bitów, 3 oznacza 8 bitów).
AH = 1 — wysłanie znaku z rejestru AL
AH = 2 — odebranie znaku
AH = 3 — odczytanie stanu
DX — urządzenie COM1 lub COM2 (0 lub 1).
Po wykonaniu funkcji do rejestru AH wpisywany jest stan linii układu 8250:
bit 7 — błąd przeterminowania (ang. time-out),
bit 6 — rejestr TSR układu 8250 jest wolny,
bit 5 — rejestr THR układu 8250 jest wolny,
bit 4 — rozpoznano sygnał BREAK,
bit 2 — błąd parzystości,
bit 1 — błąd typu Overrun,
bit 0 —dane odbierane są gotowe,
Do rejestru AL wpisywany jest stan modemu (tylko dla funkcji 3).
INT 15 — magnetofon kasetowy (tylko dla IBM PC)

Funkcja ta nie występuje w IBM PC/XT, a jej obsługa została zastąpiona przez podprogram, który zawsze generuje komunikat błędu INVALID COMMAND (AH = 96, przeniesienie ustawione)
INT 16 — odczyt klawiatury
AH = 0 — odczyt znaku z bufora klawiatury
AL — znak ASCII
AH — numer klawisza (ang. scan-code)
AH = 1 — stan bufora klawiatury
Wskaźnik Z = 0 oznacza, że znak jest gotowy do odczytania. Znak i numer klawisza znajdują się w rejestrze AX (tak jak dla funkcji 0), jednak nie są wymazane z bufora klawiatury.
AH = 2 — odczyt stanu klawiszy nieznakowanych do rejestru AL:
bit 7 — INSERT jest aktywny,
bit 6 — CAPS-LOCK jest aktywny,
bit 5 — NUM-LOCK jest aktywny,
bit 4 — SCROLL-LOCK jest aktywny,
bit 3 — przycik ALT jest wciśnięty,
bit 2 — przycisk CTRL jest wciśnięty,
bit 1 — lewy przycisk SHIFT jest wciśnięty,
bit 0 — prawy przycisk SHIFT jest wciśnięty.
INT 17 — interface drukarki
AH = 0 — drukowanie znaku z rejestru AL
AH = 1 — inicjalizacja interface'u
AH = 2 — odczyt stanu drukarki do rejestru AH:
bit 7 — nie zajęta,
bit 6 — potwierdzenie,
bit 5 — brak papieru,
bit 4 — wybranie drukarki,
bit 3 — błąd wejścia-wyjścia,
bit 2 — nieużywany,
bit 0 — błąd przeterminowania.
Do rejestru DX należy wpisać numer urządzenia (0—2, dla LPT1, LPT2 lub LPT3).
INT 18 — ROM-Basic (tylko dla IBM PC)

Uruchomienie interpretera Basica zapisanego w pamięci ROM od adresu F600H:0000H.
INT 19 — program samoładujący
Kod ścieżki 0 sektora 0 dysku A zostaje załadowany do pamięci pod adres 0000H:7C00H i następuje wykonanie programu samoładującego (ang. bootstrap).
INT 1A — zegar dzienny
Zegar dzienny działa jako czterobajtowy licznik z częstotliwością 18,2 Hz.
AH = 0 — odczyt czasu
CX — bardziej znaczące słowo licznika
DX — mniej znaczące słowo licznika
AL = 0 — brak 24-godzinnego przepełnienia od chwili ostatniego odczytu
AL>0 — nowy dzień od chwili ostatniego odczytu
AH = 1 — ustawianie czasu
CX — bardziej znaczące słowo licznika
DX — mniej znaczące słowo licznika
INT 1B — klawisz BREAK
Przerwanie jest generowane przez wciśnięcie klawiszy CTRL-BREAK. Odpowiedni wektor przerwania jest ustawiony na podprogram pusty, lecz może zostać ustawiony na adres procedury użytkownika.
INT 1C — czasomierz
Dla każdego impulsu zegara systemowego (przerwanie sprzętowe INT 8) generowane jest również przerwanie programowe INT 1C. Tak jak w wypadku INT 1B, wektor przerwania jest ustawiony przez BIOS na podprogram pusty, lecz może zostać ustawiony na procedurę użytkownika.
INT 1D — inicjalizacja ekranu monitora
Wektor tego przerwania wskazuje tablicę inicjalizacyjną dla przerwania INT 10 i funkcji 0 (wybieranie trybu pracy ekranu monitora), lecz może zostać ustawiony na tablicę zdefiniowaną przez użytkownika.
INT 1E — parametry dysku
Wektor tego przerwania wskazuje bieżący blok parametrów dysku. W wypadku użycia dysków niestandardowych lub dla innego formatu zapisu można ustawić powyższy wektor na alternatywny blok parametrów.
INT 1F — zbiór znaków graficznych
Wektor tego przerwania zawiera wskaźnik zbioru znaków ASCII, używanego w trybie graficznym. BIOS ustawia ten wskaźnik na 0:0, lecz można go zmienić odpowiednim programem (np. GRAFTABL.COM).

Na zakończenie warto podkreślić, że MS-DOS był systemem operacyjnym pierwotnie niezależnym od procedur i sprzętu, więc przedstawione wywołania systemowe BIOS-a nie muszą występować we wszystkich komputerach z tym systemem, lecz z pewnością występują w komputerach całkowicie kompatybilnych z IBM PC/XT.

Polskie nazewnictwo znaków kodu ASCII/ISO (2)

(dokończenie ze str. 31)

angielskie: substitute i delete. W trzech wypadkach propozycje nowych nazw wiążą się z rozwojem techniki komputerowej; uważam, że obecnie należałoby mówić: **sygnał dźwiękowy, wysuw strony i powrót karetki,** zamiast — dzwonek, zmiana formularza i powrót wózka.

Przy tej okazji warto też wyjaśnić zasadę pisania ciągów kodowych, odpowiadających poszczególnym znakom (a także liczbom). Otóż w języku potocznym, tj. naturalnym, etnicznym, ciąg kodowy odpowiadający określonemu znakowi może być rozumiany tylko jako przedstawiony w zapisie dziesiętnym, ponieważ na co dzień używamy dziesiętnego systemu liczenia. Dlatego, aby uniknąć nieporozumień czy dany znak jest przedstawiony w zapisie (kodzie) dwójkowym, ósemkowym, szesnastkowym czy dziesiętnym, należy przyjąć zasadę, że każdy ciąg kodowy powinien być zakończony odpowiednią literą alfabetu, oznaczającą rodzaj kodu (np. H — kod szesnastkowy, B — kod dwójkowy, S — kod ósemkowy), chyba że jest to kod dziesiętny. Tak więc ciągi kodowe w kodzie dziesiętnym, i tylko takie, jako powszechnie używane w języku naturalnym (np. do przedstawiania liczb), można zapisywać bez litery symbolizującej rodzaj kodu. Postępując zgodnie z tą zasadą, napotkawszy ciąg kodowy 11, z pewnością nie zinterpretujemy go mylnie, jako znak ETX (bo nie jest to 11B), ani jako HT (bo nie jest to 11S), ani jako DC1 (bo nie jest to 11H), lecz jako VT (bo jest zapisany w kodzie dziesiętnym).

JANUSZ ZALEWSKI

Porównanie systemów BIOS dla rodziny komputerów IBM PC

Komputery osobiste IBM PC w wersji podstawowej rozrosły się na całą rodzinę wzajemnie powiązanych mikrokomputerów. Oprócz systemu podstawowego wprowadzono takie odmiany, jak: PC/XT, XT/370, 3270 PC, wersję przenośną, PC jr, PC/AT, AT/370, a także kilka odmian XT w wersji przemysłowej i wojskowej.

Wraz z rozwojem rodziny wprowadzano lub usuwano niektóre składniki wyposażenia sprzętowego tych komputerów. W niniejszym artykule porównano różne cechy odpowiedzialnego za obsługę sprzętu systemu **BIOS** (ang. basic input-output system) dla poszczególnych przedstawicieli całej rodziny (tabele 1a—1c). Programując na określonym rodzaju sprzętu można wskazać na podstawie tego porównania, jakie funkcje są używane przez wszystkie komputery w tej rodzinie. Modele XT/370 i 3270 PC używają BIOS-a modelu PC/XT, a model AT/370 — BIOS-a modelu PC/AT.

Celem systemu **BIOS** jest zapewnienie wspólnego sprzężenia ze sprzętem dla wszystkich programów, od programów aplikacyjnych do systemów operacyjnych, tak aby zminimalizować rozmiar kodu niezbędnego do napisania przy przenoszeniu oprogramowania między komputerami. Dzięki temu **BIOS** umożliwia programistom ograniczenie zależności sprzętowych do kilku elementarnych podprogramów. Tak więc, **BIOS** zapewnia przenośność oprogramowania i kompatybilność różnych środowisk sprzętowych, pozwalając jednocześnie na zachowanie prawie pełnej szybkości i możliwości bezpośredniego dostępu do sprzętu.

Tabela 1a. Wywołania systemowe dla rodziny IBM PC (cz. I)

Numer przerwania	PC	PC/XT	PC jr	PC/AT	Uwagi
0	+	+	+	+	Dzielenie przez zero
1	+	+	+	+	Praca krokowa (używane przez DEBUG)
2	+	+	+	+	Przerwanie niemaskowalne — kontrola parzystości i przerwanie koprocesora (dla PC jr tylko przerwanie od klawiatury)
3	+	+	+	+	Punkt wstrzymania (używane przez DEBUG)
4	+	+	+	+	Nadmiar
5	+	+	+	+	Drukowanie obrazu z ekranu (PRT-SC)
6					Zarezerwowane
7					Zarezerwowane
8	+	+	+	+	Obsługa przerwania czasomierza
9	+	+	+	+	Obsługa przerwania z klawiatury
A	—	—	—	—	Kontrola ekranu, tzw. vertical retrace (rozszerzenie BIOS-a dla karty EGA)
B	+	+	+	+	Wejście sprzętowe obsługi sterownika komunikacyjnego
C	+	+	+	+	Wejście sprzętowe obsługi sterownika komunikacyjnego
D	+	+	+	+	Obsługa drugiej drukarki (dla PC/AT — przerwanie od koprocesora 80287)
E	+	+	+	+	Wejście obsługi sterownika dysku
F					Zarezerwowane

Tabela 1b. Wywołania systemowe dla rodziny IBM PC (cz. II)

Przerwanie i kod funkcji	PC	PC/XT	PC jr	PC/AT	Uwagi
10	+	+	+	+	**Monitor ekranowy**
0	+	+	+	+	Ustawienie trybu pracy (rozszerzenie BIOS-a dla karty EGA)
1	+	+	+	+	Ustawienie rodzaju kursora
2	+	+	+	+	Ustawienie położenia kursora
3	+	+	+	+	Odczyt położenia kursora
4	+	+	+	+	Odczyt położenia pióra świetlnego
5	+	+	+	+	Wybór aktywnej strony ekranu
6	+	+	+	+	Przesuwanie strony aktywnej do góry
7	+	+	+	+	Przesuwanie strony aktywnej na dół
8	+	+	+	+	Odczyt znaku i atrybutu w bieżącym położeniu kursora
9	+	+	+	+	Zapis znaku i atrybutu w bieżącym położeniu kursora
A	+	+	+	+	Zapis samego znaku w bieżącym położeniu kursora
B	+	+	+	+	Ustawianie palety kolorów
C	+	+	+	+	Zapis punktu graficznego (piksela)
D	+	+	+	+	Odczyt punktu graficznego (piksela)
E	+	+	+	+	Zapis znaków dalekopisu
F	+	+	+	+	Udostępnienie bieżącego trybu pracy
10	—	—	+	—	Ustawienie rejestrów palety kolorów (rozszerzenie BIOS-a dla karty EGA)
11	—	—	—	—	Podprogram generatora znaków (rozszerzenie BIOS-a dla karty EGA)
12	—	—	—	—	Alternatywny wybór (rozszerzenia BIOS-a dla karty EGA)
13	—	—	—	+	Zapis ciągu znaków z opcjonalnymi atrybutami (rozszerzenie BIOS-a dla karty EGA).
FE	—	—	—	—	Pobranie adresu bufora ekranu w trybie tekstowym (rozszerzenie BIOS-a dla programu Top View)
FF	—	—	—	—	Uaktualnienie ekranu w trybie tekstowym (rozszerzenie BIOS-a dla programu TopView)
11	+	+	+	+	**Określenie konfiguracji i stanu urządzeń**
	+	+	+	+	Użycie obszaru danych BIOS-a — — EQUIP-FLAG
12	+	+	+	+	**Określenie rozmiaru pamięci**
13	+	+	+	+	**Dysk elastyczny**
0	+	+	+	+	Zerowanie sterownika dysku elastycznego
1	+	+	+	+	Odczyt stanu ostatniej operacji dysku elastycznego
2	+	+	+	+	Odczyt sektorów dyskietki
3	+	+	+	+	Zapis sektorów dyskietki
4	+	+	+	+	Weryfikacja sektorów dyskietki
5	+	+	+	+	Formatowanie ścieżki dyskietki
15	—	—	—	+	Określenie rodzaju dysku (brak, elastyczny, elastyczny czuły na zmianę dyskietki, stały)
16	—	—	—	+	Zmienna stanu dysku
17	—	—	—	+	Ustalenie rodzaju dyskietki przed sformatowaniem (gęstość)
	+	+	+	+	Użycie obszaru danych BIOS-a — DISK-POINTER
	—	—	—	+	Użycie kodu sterującego dysku (ang. state machine) — komórki 40:90-40:95

Przerwanie i kod funkcji	PC	PC/XT	PCjr	PC/AT	Uwagi
13		+		+	**Dysk stały** (jeżeli zainstalowany jest dysk stały, to przerwanie INT 13 jest przeadresowane na INT 40 i wskazuje punkt wejściowy BIOS-a do obsługi dysku stałego
0		+		+	Zerowanie sterownika dyskowego
1		+		+	Odczyt stanu ostatniej operacji
2		+		+	Odczyt sektorów
3		+		+	Zapis sektorów
4		+		+	Weryfikacja sektorów
5		+		+	Formatowanie ścieżki
6		+		—	Formatowanie ścieżki i ustawienie wskaźników uszkodzonych sektorów
7		+		—	Formatowanie od określonej ścieżki
8		+		+	Odczytanie aktualnych parametrów dysku
9		+		+	Inicjalizacja parametrów dysków
A		+		+	Odczyt długich sektorów dysku 20 MB
B		+		+	Zapis długich sektorów dysku 20 MB
C		+		+	Poszukiwanie określonego cylindra
D		+		+	Zerowanie drugiego sterownika dyskowego
E		+		—	Odczyt bufora sektorów
F		+		—	Zapis bufora sektorów
10		+		+	Badanie gotowości napędu
11		+		+	Kalibracja
12		+		—	Diagnostyka pamięci RAM sterownika
13		+		—	Diagnostyka napędu
14		+		+	Wewnętrzna diagnostyka sterownika
15		—		+	Określenie rodzaju dysku (brak, elastyczny, elastyczny czuły na zmianę dyskietki, stały)
14	+	+	+	+	**Łącze szeregowe**
10	+	+	+	+	Inicjalizacja łącza szeregowego
1	+	+	+	+	Nadawanie znaku
2	+	+	+	+	Odbior znaku
3	+	+	+	+	Pobranie stanu łącza szeregowego
	+	+	+	+	Użycie obszaru danych BIOS-a RS232_ BASE (0:3) — miejsce dla czterech łączy lecz BIOS wspomaga użycie tylko dwóch
	+	+	+	+	Użycie obszaru danych BIOS-a RS232_ _TIM_OUT
15	+		+		**Pamięć kasetowa lub nowe funkcje** (funkcje oznaczone gwiazdką nie są funkcjami BIOS-a w ścisłym sensie lecz stanowią wywołania systemu operacyjnego)
0	+		+		Włączenie silnika
1	+		+		Wyłączenie silnika
2	+		+		Odczyt z kasety
3	+		+		Zapis na kasetę
20*	—	—	—	+	Rozpoczęcie (AL = 10) lub zakończenie (AL = 11) programu SYSREQ
80*	—	—	—	+	Otwarcie urządzenia
81*	—	—	—	+	Zamknięcie urządzenia
82*	—	—	—	+	Zakończenie programu urządzenia
83	—	—	—	+	Oczekiwanie na zdarzenie
84	—	—	—	+	Obsługa drążka sterowniczego
85*	—	—	—	+	Obsługa klawisza zgłoszenia systemo-, wego (AL = 0 — przyciśnięcie, AL = 1 — zwolnienie)
86	—	—	—	+	Oczekiwanie przez N mikrosekund (CX,DX = N)
87	—	—	—	+	Przesyłanie blokowe (obejmuje także pamięć rozszerzoną)
88	—	—	—	+	Określenie rozmiaru rozszerzonej pamięci
89	—	—	—	+	Przełączenie procesora na tryb wirtualny
90*	—	—	—	+	Pętla zajętości urządzenia
91*	—	—	—	+	Ustawienie bitu flagowego i zakończenie przerwania

Przerwanie ikodfunkcji	PC	PC/XT	PCjr	PC/AT	Uwagi
16	+	+	+	+	**Klawiaitura**
0	+	+	+	+	Odczyt kolejnego znaku z klawiatury
1	+	+	+	+	Ustawenie bitu flagowego Z przy zajętości bufora
2	+	+	+	+	Odczytanie stanu klawiszy pomocniczych
3	—	—	+	—	Ustawienie parametrów klawiatury Typematic
4	—	—	+	—	Wyłączenie sygnalizacji dźwiękowej uderzenia w klawisze
17	+	+	+	+	**Drukarka**
0	+	+	+	+	Wysłanie znaku na drukarkę
1	+	+	+	+	Inicjalizacja drukarki
2	+	+	+	+	Odczytanie stanu drukarki
	+	+	+	+	Użycie obszaru danych BIOS-a — PRINTER_BASE (0:2); miejsce dla trzech drukarek
	+	+	+	+	Użycie obszaru danych BIOS-a — PRINTER_TIM_OUT
18	+	+		+	**Uaktywnienie Basica w pamięci ROM**
19	+	+	+	+	**Inicjowanie programu samoladującego**
1A	+	+	+	+	**Czas dzienny**
0	+	+	+	+	Odczyt liczby zliczeń
1	+	+	+	+	Ustawienie liczby zliczeń
2	—	—	—	+	Odczyt czasu dziennego
3	—	—	—	+	Ustawienie czasu dziennego
4	—	—	—	+	Odczyt daty
5	—	—	—	+	Ustawienie daty
6	—	—	—	+	Ustawienie alarmu (powstanie alarmu wywołuje przerwanie INT 4A)
7	—	—	—	+	Skasowanie alarmu
1B	+	+	+	+	**Obsługa klawisza CTRL-BREAK**
1C	+	+	+	+	**Programowe zliczanie impulsów**
1D	+	+	+	+	**Tablica parametrów ekranu**
1E	+	+	+	+	**Tablica parametrów dysku**
1F	+	+	+	+*	**Tablica znaków graficznych** (dla grafiki o dużej rozdzielczości)

BIOS składa się z podprogramów zapewniających, na poziomie kodu maszynowego, obsługę głównych urządzeń we-wy znajdujących się w systemie. W komputerach rodziny IBM PC, **BIOS** jest zawarty w pamięci ROM na płycie systemowej, razem z interpreterem Basica i zbiorem podprogramów diagnostycznych POST (ang. power on self test), służących do testowania komputera po włączeniu zasilania.

BIOS zapewnia niezależność sprzętową dzięki wprowadzeniu poziomu pośredniego, oddzielającego oprogramowanie od sprzętu. Przykładowo, używając wywołania BIOS-a do przesłania znaku na drukarkę, programista nie musi wiedzieć, jaki jest adres portu drukarki, ani jak nim sterować.

BIOS jest wywoływany przez odpowiednie przerwania wektorowe do jego poszczególnych punktów wejściowych (ang. entry points). Inne wektory przerwań służą do obsługi przerwań sprzętowych, jak np. zakończenie operacji dyskowej (tabela 2). W praktyce, programowe wywołanie BIOS-a polega na załadowaniu odpowiednich rejestrów i wykonaniu instrukcji INT z właściwym argumentem. Przykładowy fragment kodu takiego wywołania przedstawiono na wydruku.

```
MOV  AH,0     ;Załaduj do rejestru AH kod funkcji
              ;drukowania znaku zapisanego w AL
MOV  AL,'B'   ;Wpisz do rejestru AL znak
              ;przeznaczony do drukowania, np. 'B'
MOV  DX,0     ;Numer urządzenia drukującego, np.
              ;LPT1, wpisz do rejestru DX
INT  17       ;Wywołaj obsługę drukarki przez BIOS
```

Przykładowy fragment kodu wywołania przerwania programowego

Instrukcja INT (tj. przerwanie programowe) powoduje przekazanie sterowania mikroprocesorem do podprogramu, którego adres znajduje się w 4-bajtowym wektorze odpowiadającym numerowi przerwania [1]). Ogółem mikrokomputery tej rodziny używają 256 przerwań, przy czym pierwszych 128 jest przeznaczonych dla **BIOS-a** i systemu o-

[1]) Por. „Podprogramy obsługi przerwań dla IBM PC", Informatyka nr 7—8, str. 48—50, 1986.

peracyjnego, a pozostałe — dla interpretera Basica. Odpowiadające tym przerwaniom 256 adresów wektorów zajmuje pierwszy kilobajt pamięci, w taki sposób, że bajty 0—3 odpowiadają przerwaniu INT 0, a bajty 3FCH—3FFH — przerwaniu INT FFH. Wszystkie przerwania modelu PC/AT są ważne tylko w trybie rzeczywistym (ang. real mode).

System **BIOS** jest rozszerzalny (tabela 3). Podprogramy diagnostyczne POST, wykonywane po włączeniu zasilania, m.in. sprawdzają cały obszar adresowy pamięci ROM w poszukiwaniu dodatkowych programów, które instalują się same. Przykładem takiego rozszerzenia jest karta EGA (ang. extended graphics adapter), która rozszerza przerwanie programowe INT 10, w sposób przedstawiony w tabeli 1 (cz. 2).

W celu samoczynnego zainstalowania się jako rozszerzenie BIOS-a, podprogram powinien wykonać następujące czynności:

Tabela 1c. Wywołania systemowe dla rodziny IBM PC (cz. III)

Numer przerwania	PC	PC/XT	PC jr	PC/AT	Uwagi
20-3F					Funkcje systemu PC-DOS
40	—	+	—	+	Wskaźnik wejścia BIOS-a do obsługi dysku
41	—	+	—	+	Wskaźnik bloku parametrów pierwszego dysku stałego
42	—	—	—	—	Wskaźnik wejścia BIOS-a do obsługi ekranu (rozszerzenie BIOS-a dla karty EGA)
43	—	—	—	—	Wskaźnik parametrów inicjalizacji karty EGA (rozszerzenie BIOS-a)
44	—	—	+	—	Wskaźnik tablicy znaków graficznych karty EGA (rozszerzenie BIOS-a; stosowane niezależnie w PC jr)
45					Zarezerwowane
46	—	—	—	+	Wskaźnik parametrów drugiego dysku stałego
47					Zarezerwowane
48	—	—	+	—	Użycie klawiatury bez sznura
49	—	—	+	—	Adres tablicy numerów znaków nie objętych klawiaturą
4A-59					Zarezerwowane
5A					Adres wejścia BIOS-a do obsługi adaptera klastra (rozszerzenie BIOS-a)
5B					Zarezerwowane
5C	—	—	—	—	Punkt wejściowy NETBIOS-a do obsługi sieci (rozszerzenie BIOS-a)
5D-5F					Zarezerwowane
60-67					Zarezerwowane dla przerwań programów użytkowych
68-6F					Nieużywane
70	—	—	—	+	IRQ8 — przerwanie zegarowe
71	—	—	—	+	IRQ9 — przeadresowanie na IRQ2
72	—	—	—	+	IRQ10
73	—	—	—	+	IRQ11
74	—	—	—	+	IRQ12
75	—	—	—	+	IRQ13 — przerwanie koprocesora, przeadresowane przez BIOS na INT2 (NMI)
76	—	—	—	+	IRQ14 — przerwanie od dysku stałego
77	—	—	—	+	IRQ15
78-7F					Niużywane
80-85					Zarezerwowane dla Basica
86-F0					Używane przez interpreter Basica
F1-FF					Nieużywane

Tabela 2. Przerwania sprzętowe dla komputerów rodziny IBM PC

Linia	PC i PC/XT	PC jr	PC/AT
NMI	Błędy parzystości	Przerwanie od klawiatury	Błędy parzystości
IRQ0	Czasomierz	Przerwanie czasomierza i zegara	Wyjście 0 czasomierza
IRQ1	Klawiatura	Kanał we-wy (zarezerwowane)	Klawiatura (wypełnienie bufora)
IRQ2	Zarezerwowane	Kanał we-wy	Dołączenie IRQ8—IRQ15
IRQ3	Port szeregowy nr 2	Port RS-232C	Port szeregowy nr 2
IRQ4	Port szeregowy nr 1	Modem	Port szeregowy nr 1
IRQ5	Dysk stały (nie dotyczy PC)	Kontrola ekranu (tzw. vertical retrace)	Port równoległy nr 2
IRQ6	Sterowanie dyskiem elastycznym	Dysk elastyczny	Sterowanie dyskiem elastycznym
IRQ7	Port równoległy	Kanał we-wy (drukarka równoległa)	Port równoległy nr 1
IRQ8	—	—	Zegar
IRQ9	—	—	Przeadresowane na IRQ2
IRQ10	—	—	Zarezerwowane
IRQ11	—	—	Zarezerwowane
IRQ12	—	—	Zarezerwowane
IRQ13	—	—	Koprocesor
IRQ14	—	—	Sterownik dysku stałego
IRQ15	—	—	Zarezerwowane

- skopiować bieżącą wartość wskaźnika przerwań,
- zablokować przerwania,
- zastąpić bieżącą wartość wskaźnika przerwań adresem wejściowym podprogramu,
- odblokować przerwania.

Jeżeli podprogram instaluje się po zainstalowaniu systemu DOS, to można używać funkcji systemowych: 35 (pobierz wektor przerwania) i 25 (ustaw wektor przerwania).

Jeżeli zainstalowane jest rozszerzenie NETBIOS, to wykonuje on przerwania 13 i 17, a także 2 lub 3. Ponadto

Tabela 3. Adresy rozszerzań BIOS-a

Zakres adresów	Wielkość obszaru	Przeznaczenie
C0000-C3FFF	16 KB	Karta EGA
C4000-C5FFF		—
C6000-C63FF	256 B	Obszar komunikacyjny profesjonalnego monitora graficznego
C6400-C7FFF		—
C8000-CBFFF	16 KB	Dysk stały
CC000-CDFFF	8 KB	NETBIOS
CE000-CFFFF		—
D0000-D7FFF	32 KB	Adapter klastra
D8000-DBFFF		—
DC000-DFFFF		—
E0000-E3FFF		—
E4000-E7FFF		—
E8000-EBFFF		
EC000-EFFFF		—
F0000-FFFFF	64 KB	Interpreter Basica i uproszczona BIOS

przerwanie INT 18 jest przeadresowane na INT 86, a funkcje WAIT i POST przerwania 15 są rozszerzone.

Funkcja WAIT (A = 90) jest używana przez dyspozytor zadań systemu operacyjnego w celu przydzielenia procesora innemu zadaniu, gdy bieżące zadanie oczekuje na zakończenie operacji wejścia-wyjścia. Jest to najbardziej efektywna postać wielozadaniowości.

Funkcja POST (A = 91) sygnalizuje zakończenie operacji wejścia-wyjścia i służy do zawiadomienia dyspozytora zadań systemu operacyjnego o zakończeniu tej operacji dla zadania oczekującego i o konieczności przesunięcia tego zadania do kolejki zadań gotowych.

Zasadą obowiązującą w BIOS-ie jest użycie jednego przerwania programowego na urządzenie. Dodatkowo można też używać jednego lub więcej wejść sprzętowych oraz jednego lub kilku wejść wskazujących tablice lub bloki danych programu obsługi urządzenia (ang. device driver). Użycie wektorów przerwań jako wskaźników danych zamiast kodu, umożliwia łatwe modyfikowanie środowiska, jak np. zbioru wyświetlanych znaków o kodach 80H-FFH dla karty grafiki kolorowej.

Według firmy IBM, jedynym sposobem bezpiecznego ominięcia BIOS-a jest dostęp do następujących portów we-wy:
- 21H — rejestr maski przerwań,
- 40H-42H — czasomierz (bez zmiany zawartości portu 41H); częstotliwość zegarowa pozostaje stała, równa 1,19 MHz,
- 61H — sterowanie dźwiękiem,
- 201H — adapter do gier (przy użyciu czasomierza do generowania opóźnień).

Używając bezwzględnych adresów pamięci, należy pamiętać o pewnych szczegółach. Dla wektorów przerwań (od 0:0 do 0:3FF) można dodać nowe funkcje lecz nie można przedefiniować istniejących. Dla buforów ekranu monitora (B000:0 i B800:0) mapa obrazu nie zmienia się przy określonym przez BIOS trybie pracy. Jeżeli mapa bitowa ulegnie zmianie, to definiuje się nowy tryb pracy. Dla obszaru danych BIOS-a, zaczynającego się od adresu 40:0, na ogół zakłada się, że zmienne zachowują swoje definicje dopóty, dopóki zdefiniowane są odpowiednie funkcje, lecz nie należy temu dawać wiary.

Opracował: JANUSZ ZALEWSKI
na podst.: BYTE, No. 10, 1985

Szczególnie istotne adresy, przerwania i porty IBM PC/XT

Wobec niedokładności danych zawartych w literaturze firmowej i w związku z trudnościami w dotarciu do tej literatury, publikujemy informacje dotyczące podziału pamięci, obsługi przerwań i najważniejszych portów w systemach PC/XT. Podając te dane warto jednak przestrzec mniej doświadczonych programistów przed samodzielnym eksperymentowaniem ze sprzętem i oprogramowaniem systemowym, gdyż może to niekiedy spowodować nieprzewidziane skutki.

W uzupełnieniu informacji o obsłudze przerwań warto dodać, że układ przerwań, zaprojektowany w oparciu o sterownik 8259A-5, może przyjmować sygnały przerwań z ośmiu źródeł. Zgłoszenie przerwania jest dokonywane przez przełączenie stanu na linii wejściowej. Po wykryciu zgłoszenia, wewnętrzny układ priorytetowy wyznacza jeszcze nie obsługiwane przerwanie o najwyższym priorytecie i zgłasza je do procesora. Procesor, o ile aktualnie może przyjmować przerwania, odczytuje adres i wykonuje skok ze śladem do procedury obsługi przerwania. Na końcu procedury obsługi przerwania, do sterownika przerwań wysyłany jest rozkaz informujący go o zakończeniu obsługi danego przerwania. Przerwaniom sprzętowym w komputerach IBM PC/XT odpowiadają przerwania programowe: od INT 08H do INT 0FH.

Tabela 1. Szczególnie istotne obszary pamięci

Obszar	Funkcja
0:400-48F	BIOS
490-4EF	Zarezerwowane
4F0-4FF	Obszer komunikacyjny
500-5FF	DOS/BASIC
500	Wskaźnik PRINT SCREEN, 0 — nieaktywny, 1 — aktywny, FF — błąd
504	Bajt stanu napędu dyskowego
00000-3FFFF	Pamięć RAM na płycie głównej
40000-9FFFF	Pamięć kanału we-wy (może być na płycie głównej)
A0000-AFFFF	Zarezerwowane dla płyty wideo
B0000-B3FFF	Ekran monochromatyczny
B4000-B7FFF	Zarezerwowane dla ekranu monochromatycznego
B8000-BBFFF	Ekran kolorowy
BC000-BFFFF	Zarezerwowane dla ekranu kolorowego
C0000-C7FFF	Zarezerwowane
C8000-CBFFF	Pamięć ROM dysku Winchester
CC000-CFFFF	Pamięć ROM nakładek BIOS
D0000-DFFFF	Pamięć ROM dla płytki Z80-IO
F000:E000-F000:FFFF	ROM BIOS

Tabela 2. Podział przestrzeni wejścia-wyjścia

Obszar adresowy	Urządzenie
000-00F	8237A-5 (DMA)
020-021	8259A-5 (sterownik przerwań)
040-043	8253-5 (CTC)
060-063	8255A-5 (PIO)
080-083	74LS670 (rejestry stron DMA)
0A0-0AF	Maska NMI
0C0-0CF	—
0E0-0EF	—
200-20F	Adapter do gier
210-217	Blok rozszerzenia
220-24F	—
240-24F	Zegar nr 2
278-27F	—
2F0-2F7	—
2F8-2FF	Sterownik komunikacji asynchronicznej nr 2
300-31F	Płytka prototypowa
320-32F	Dysk stały
340-34F	Zegar nr 1
378-37F	Drukarka
380-38C	Sterownik SDLC
380-38F	Procesor Z80-IO (CS88-Z80)
380-389	Sterownik BSC nr 2
3A0-3A9	Sterownik BSC nr 1
3B0-3BF	Monitor monochromatyczny (Hercules)
3C0-3CF	—
3D0-3DF	Monitor kolorowy (CGA)
3E0-3F0	—
3F0-3F7	Dysk elastyczny
3F8-3FF	Sterownik komunikacji asynchronicznej nr 1

Tabela 3. Obsługa przerwań dla IBM PC/XT

Numer przerwania	Adres wektora	Nazwa	Adres obsługi	Obsługuje
INT 0	0000	D11	00E3:3143	DOS
INT 1	0004	D11	0070:013F	CSM
INT 2	0008	NMI-INT	?	DOS/BIOS
INT 3	000C	D11	0070:013F	CSM
INT 4	0010	D11	0070:013F	CSM/DOS
INT 5	0014	PRINT--SCREEN	F000:FF54	BIOS
INT 6	0018	D11	F000:FF23	BIOS (?)
INT 7	001C	D11	F000:FF23	BIOS (?)
INT 8	0020	TIMER-INT	F000:FEA5	BIOS/DOS
INT 9	0024	KB-INT	F000:E987	BIOS/DOS
INT A	0028	D11	F000:FF23	BIOS (?)
INT B	002C	D11		BIOS (?)/program
INT C	0030	D11		BIOS (?)/program
INT D	0034	D11	C800:01BE	BIOS
INT E	0038	DISK-INT	F000:EF57	BIOS
INT F	003C	D11	0070:013F	BIOS/spooler/CSM
INT 10	0040	VIDEO-IO	F000:F065	BIOS
INT 11	0044	EQUIPMENT	F000:F84D	BIOS
INT 12	0048	MEMORY-SIZE	F000:F841	BIOS
INT 13	004C	DISKETTE-IO	C800:01CF/F000:EC59	BIOS
INT 14	0050	RS232-IO	F000:E739	BIOS
INT 15	0054	CASETTE-IO	F000:F859	BIOS
INT 16	0058	KEYBOARD--IO	F000:E82E	BIOS
INT 17	005C	PRINTER-IO	F000:EFD2	BIOS
INT 18	0060	BASIC	F600:0000/F000::FF23	BIOS/Basic
INT 19	0064	BOOT-STRAP	C800:014E/F000:E6F1	BIOS
INT 1A	0068	TIME-OF-DAY	F000:FE6E	BIOS
INT 1B	006C	DUMMY--RETURN	0070:0138	DOS/CSM
INT 1C	0070	DUMMY-RETURN	F000:FF53	Program/BIOS
INT 1D	0074	VIDEO-PARMS	F000:F0A4	BIOS/DOS
INT 1E	0078	DISK-BASE	0000:0522	BIOS/DOS
INT 1F	007C		0000:0000	BIOS
INT 20	0080		00E3:0BFB	DOS
INT 21	0084		00E3:0BFF	DOS/CSM
INT 22	0088			DOS
INT 23	008C			DOS
INT 24	0090			DOS
INT 25	0094			DOS
INT 26	0098			DOS
INT 27	009C			DOS
INT 28-3F	00A0	Rezerwa DOS		
INT 40-5F	0100	Rezerwa		
INT 60-67	0180	Rezerwa dla użytkownika		
INT 68-7F	01A0	Niewykorzystane		
INT 80-85	0200	Rezerwa dla Basica		
INT 86-F0	0218	Używane przez Basic		
INT F1-FF	03C4--03FF	Niewykorzystane		

Opracował:
WALDEMAR DWORAKOWSKI

Ogłoszenia • Ogłoszenia • Ogłoszenia • Ogłoszenia

BIURO USŁUG KOMPUTEROWYCH. Pośrednictwo sprzedaży mikrokomputerów, części zamiennych. Warszawa, tel. 41-44-48.

EO/272/K/86

Ogłoszenia • Ogłoszenia • Ogłoszenia • Ogłoszenia

Nowe książki

Niemiecka literatura informatyczna

Współczesna fachowa literatura informatyczna w języku niemieckim jest coraz liczniejsza i wzbogaca corocznie światowe zestawienia bibliograficzne w tej dziedzinie o kilkaset doskonałych pozycji. Należy z całym obiektywizmem stwierdzić, że literatura ta podtrzymuje znakomite tradycje niemieckiego piśmiennictwa naukowego i technicznego i wyróżnia się wśród literatury innych krajów precyzją, systematycznością i klarownością prezentacji tematów.

Jakkolwiek postęp w dziedzinie informatyki płynął i nadal płynie w głównej mierze z obszaru języka angielskiego, to jednak w porównaniu z literaturą informatyczną tego obszaru opracowania niemieckie reprezentują w większości wypadków lepszy poziom przekazu informacji. Zwłaszcza w zakresie upowszechniania i pogłębiania wiedzy, a więc publikacji o charakterze podręczników, warunkujących tempo komputeryzacji w równym stopniu jak dostawy sprzętu i oprogramowania.

Współczesna fachowa literatura niemiecka jest bezwzględnie najlepsza i stanowić może wzorzec dla innych krajów. Liczebność tej literatury powoduje, że również do naszej redakcji coraz częściej docierają pozycje godne polecenia. Spośród nich w pierwszej kolejności chcielibyśmy zaprezentować trzy interesujące pozycje mało znanego jeszcze na naszym rynku wydawnictwa **expert verlag** w Sindelfingen (RFN). Na marginesie warto tu zauważyć, że właśnie w Sindelfingen od co najmniej 25 lat znajduje się największy na kontynencie europejskim ośrodek badawczo-rozwojowy firmy IBM. Fakt ten w znacznym stopniu tłumaczy doskonały poziom merytoryczny prezentowanych poniżej pozycji wspomnianego wydawnictwa. Wszystkie należą do serii wydawniczej **Kontakt & Studium** obejmującej podręczniki i książki specjalistyczne z różnych dziedzin nauki i techniki. Redaktorem i wydawcą serii jest prof. dr inż. W. S. Bartz, z ośrodka kształcenia i doskonalenia kadr przy Akademii Technicznej w Esslingen (Technische Akademie Esslingen — TAE).

Krótka charakterystyka nadesłanych do recenzji książek z dziedziny informatyki:

Scheibl H. J.: BASIC — Einführung (Wprowadzenie do języka BASIC), 1982, format A5, 248 stron
Książka jest 97 tomem wspomnianej serii wydawniczej. Oparta została na doświadczeniach kursu prowadzonego w TAE pod identyczną nazwą dla inżynierów wszystkich specjalności oraz praktyków-ekonomistów. Obejmuje kompletny opis języka, ilustrowany licznymi przykładami oraz pokazuje jego praktyczne zastosowania. Podręcznik jest przeznaczony dla pracowników komórek informatycznych i działów merytorycznych przedsiębiorstw, a także początkujących programistów.

Scheibl H. S.: BASIC — Erweiterung (Rozszerzenie języka BASIC), 1984, format A5, 329 stron
Książka jest 107 tomem wspomnianej serii wydawniczej. Stanowi logiczną kontynuację poprzedniego tytułu, koncentrując się na prezentacji metod prowadzących do uzyskania optymalnych programów. Punktem ciężkości jest opracowanie zbiorów danych ilustrowane licznymi przykładami zastosowań. Praca jest adresowana do profesjonalnych użytkowników komputerów, czym odróżnia się od większości pozycji o podobnym lub zbliżonym tytule.

Schröder K.: Unix — das System und Seine Handhabung (Unix — system i jego użycie), 1984, format A5, 272 strony
Książka jest 154 tomem wspomnianej serii wydawniczej. Zawiera genezę i filozofię systemu oraz szczegółowe omówienie jego konstrukcji, ilustrowane licznymi przykładami i zestawieniami tabelarycznymi. Jest adresowana do szerokiego kręgu profesjonalnych użytkowników komputerów, zarówno początkujących, jak i zaawansowanych w znajomości problematyki systemów operacyjnych.

W.K.

MARIA MELER-KAPICA
Instytut Okrętowy
Politechnika Gdańska

System relacyjnej bazy danych SELKO dla minikomputera MERA 400

System relacyjnej bazy danych SELKO został opracowany w Instytucie Okrętowym Politechniki Gdańskiej dla minikomputera MERA 400. Pracuje on pod nadzorem systemu operacyjnego CROOK-4. System SELKO stanowi wygodne narzędzie do szybkiego wdrażania dowolnych zastosowań. Umożliwia realizację zarówno prostych, jak i złożonych zadań użytkowych, dających się łatwo definiować.

Przyjęta forma konwersacji z systemem upraszcza jego obsługę, gdyż nie wymaga specjalnego przygotowania użytkownika z zakresu programowania. W odróżnieniu od innych systemów baz danych, w systemie SELKO nie ma potrzeby pisania programów użytkowych.

Funkcje systemu bazy danych SELKO:

- tworzenie i zakładanie plików z rekordami o następujących typach pól: alfanumerycznym, alfabetycznym, numerycznym bez możliwości wykonywania działań arytmetycznych, numerycznym z deklaracją formatu, dat;
- wprowadzanie danych z automatyczną kontrolą ich poprawności formalnej w zależności od typu pola i innych parametrów;
- aktualizacja danych z możliwością dopisywania danych lub ich modyfikacji;
- sortowanie plików według dowolnych pól lub części pól;
- selekcja rekordów bezpośrednio według wartości pól lub według innych wartości, obliczonych na ich podstawie;
- rzutowanie plików, tj. tworzenie nowych plików zawierających rekordy z plików wejściowych, zredukowane do wskazanych pól. (Rzutowanie może być operacją wykonywaną samodzielnie bądź łącznie z operacją selekcji);
- generowanie złożonych raportów, m.in. wielopoziomowych tabel informacyjnych oraz statystyk;
- łączenie dwu plików o różnych strukturach rekordów, w wyniku którego otrzymuje się plik z rekordami o nowej wskazanej strukturze; w trakcie łączenia mogą być realizowane obliczenia warunkowe i podstawienia;
- przechowywanie opisów standardowych wydruków raportów;
- informowanie o aktualnej strukturze rekordów w istniejących plikach (wydruk skorowidzów plików i pól);
- kasowanie nieaktualnych opisów plików.

Metoda konstrukcji bazy danych

System SELKO jest zbiorem programów napisanych w języku BASIC z wbudowanymi procedurami asemblerowymi. Ponadto istnieją uproszczone wersje programów zapisywania i aktualizacji danych oraz sortowania plików i selekcji informacji z bazy danych, napisane w całości w języku asemblera ASSM.

SELKO jest systemem o relacyjnej strukturze plików, wyposażonym w takie relacyjne operatory algebraiczne jak: rzut (ang. projection), selekcja (ang. selection) oraz połączenie (ang. join). Oprogramowanie bazy danych ma strukturę modularną z punktu widzenia realizowanych funkcji. Konwersacyjny tryb pracy, przyjęty we wszystkich programach jest użyteczny zwłaszcza na etapie wdrażania zastosowań.

Dla ułatwienia regularnej eksploatacji systemu jest możliwe utworzenie specjalizowanego pakietu w postaci makrobloku, którego zadaniem jest automatyczne sterowanie przebiegiem rutynowego przetwarzania.

System bazy dany SELKO charakteryzują następujące cechy:

- **Prostota obsługi,** wynikająca z konwersacyjnego sposobu komunikowania się z bazą danych, eliminująca potrzebę pisania programów użytkowych.
- **Dwa tryby pracy:**
  — interakcyjny — szczególnie przydatny na etapie projektowania i wdrażania nowych zastosowań (elastyczność, łatwość zmian i reorganizacji),
  — wsadowy — w fazie eksploatacji po dopracowaniu szczegółów przetwarzania.
- **Możliwość współpracy z innymi systemami** (przekazywanie i pobieranie danych), umożliwiającej korzystanie i przetwarzanie wyników programów napisanych w innych językach programowania.
- **Możliwość deklaracji ograniczeń integralności,** tj. dopuszczalnych zakresów wartości danych i automatycznej kontroli prawidłowości wprowadzanych danych.

System bazy danych SELKO został opracowany z myślą o fakturowaniu usług oraz rozrachunków z kontrahentami Stoczni Remontowej Radunia. Znalazł on również zastosowanie w Gdańskich Zakładach UNIMOR, Elektrowni Kozienice i kilkunastu innych ośrodkach obliczeniowych w kraju: w Warszawie, Łodzi, Poznaniu, Gliwicach, Zgierzu, Raszynie. Jest sukcesywnie testowany i wdrażany w księgowości, w dziekanacie oraz w dziale inżyniera do spraw aparatury w Instytucie Okrętowym Politechniki Gdańskiej.

Duże możliwości systemu zostały sprawdzone w następujących zastosowaniach:
— rozliczenia i kontrola finansowa wykonania umów,
— sporządzanie kalendarium płatności za usługi stoczni,
— ewidencja osobowa,
— ewidencja przychodów i rozchodów aparatury,
— sporządzanie pełnej dokumentacji systemu rekrutacji na studia,
— obliczanie stypendium i sporządzanie listy płac,
— ewidencja wyników nauczania studentów wszystkich lat studiów,
— dydaktyka w Instytucie Okrętowym.

Sprostowanie

Bardzo przepraszamy pana **prof. dr. hab. inż. Ryszarda Tadeusiewicza** za mylne podanie (INFORMATYKA nr 2, 1987, s. 22) miejsca Jego pracy. Pan profesor **jest pracownikiem Instytutu Automatyki Akademii Górniczo-Hutniczej w Krakowie,** a nie — jak mylnie podaliśmy — Instytutu Informatyki AGH.

Redakcja

V A-C. Bazy danych

Samotesty, opracowane i opublikowane przez Association for Computing Machinery, nie stanowią standardu egzaminacyjnego i są przeznaczone wyłącznie do samodzielnego sprawdzania własnej wiedzy [1]).

A. Ogólne pojęcia bazy danych

PYTANIA

1. Na czym polega istotna wyższość systemów zarządzania bazami danych — SZBD (ang. Data Base Management Systems — DBMS) nad konwencjonalnymi systemami plików (ang. conventional file systems)?

a) wspólny dostęp (ang. shared access) do danych,
b) brak konieczności badania poprawności danych (ang. data validation),
c) racjonalna liczba danych nadmiarowych (ang. redundat data),
d) łatwiejsze egzekwowanie ochrony danych (ang. protection policies).

2. Należy skojarzyć wymienione pojęcia (I...IV) z odpowiednimi definicjami (a...d):

I. Język zapytań (ang. query language);
II. Język opisu danych (ang. data definition language);
III. Język manipulowania danymi (ang. data manipulation language);
IV. Język sterowania urządzeniami (ang. device and media control language);

a) nieproceduralny język wysokiego poziomu do wyszukiwania danych (ang. data retrieval) na podstawie SZBD,

b) język opisujący odwzorowanie danych na fizycznym nośniku pamięciowym (ang. mapping of data to physical storage),

c) język opisu obiektów świata rzeczywistego i ich atrybutów oraz relacji zachodzących między tymi obiektami, czyli język odzwierciedlania logicznej organizacji danych (ang. logical organization of data),

d) język niskiego poziomu do gospodarowania pojedynczymi rekordami w bazie danych (ang. traversing through the database one record at a time).

3. Niezależność danych (ang. data independence), pożądana cecha systemu zarządzania bazą danych, zakłada niewrażliwość programów aplikacyjnych na zmiany w fizycznej strukturze pamięci, a w miarę możliwości także na zmiany w logicznej strukturze danych. Które z wymienionych poniżej mechanizmów SZBD nie mają wpływu na zapewnienie niezależności danych?

a) wydzielenie schematu i podschematów w języku opisu danych,
b) wydajne systemowo mechanizmy dostępu do danych,
c) słowniki danych,
d) jednolite traktowanie danych i relacji zachodzących między nimi.

B. Relacyjne bazy danych

PYTANIA

4. Wymień wszystkie najważniejsze zalety relacyjnego modelu bazy danych:

a) niezależność danych,
b) większe możliwości sterowania pamięcią oraz urządzeniami współpracującymi,
c) głębokie podstawy matematyczne,
d) symetria wyrażeń języków zapytań.

5. Rozważmy następujące relacje w bazie danych:

ZAMÓWIENIE (nr_zamówienia, element, ilość)
KLUCZ (nr_zamówienia, element)
ZAPOTRZEBOWANIE (inwestycja, element, ilość)
KLUCZ (inwestycja, element)
DOSTAWA (nr_zamówienia, dostawca, data_zamówienia)
KLUCZ (nr_zamówienia)

Na które z poniższych pytań będzie mógł odpowiedzieć dany relacyjny system zarządzania bazą danych?

a) czy zamówiono element E4 do realizacji inwestycji I3?
b) u którego dostawcy zamówiono element E3?
c) czy dostawca D4 obsługuje inwestycję I2?
d) czy liczba zamówionych elementów E5 przekracza potrzeby inwestycji I3?

6. Załóżmy, że w relacji

R(A, B, C, D, E, F)
istnieją następujące zależności funkcjonalne:

$A \rightarrow B$	$A \rightarrow E$	$A \rightarrow E$
$AC \rightarrow D$	$CE \rightarrow D$	$D \rightarrow F$

W których z poniższych grup relacji, przedstawiających powyższą relację R, wszystkie relacje są w trzeciej formie normalnej (ang. third normal form)?

a) (A, B, C, D, E) (C, D, E, F),
b) (A, C, D) (A, B, E) (C, E, D, F),
c) (A, C, D) (A, B) (B, E) (C, E, D) (D, F),
d) (A, C, D) (A, B, E) (C, E, D) (D, F).

C. Sieciowe bazy danych

PYTANIA

7. Które z poniższych stwierdzeń są prawdziwe dla struktury danych w standardzie CODASYL/DBTG (ang. Data Base Task Group), zwanej kolekcją (ang. set)?

a) dany typ rekordu nie może być jednocześnie podrzędnym (ang. member) i nadrzędnym (ang. owner) w tym samym typie kolekcji (ang. set-type),
b) dany typ rekordu nie może być jednocześnie nadrzędnym dla dwu lub więcej różnych typów kolekcji,
c) dany typ rekordu nie może być jednocześnie podrzędnym w dwóch lub kilku różnych typach kolekcji,
d) wystąpieniu kolekcji musi odpowiadać przynajmniej jeden rekord podrzędny,
e) dany rekord może być związany tylko z jednym wystąpieniem kolekcji określonego typu.

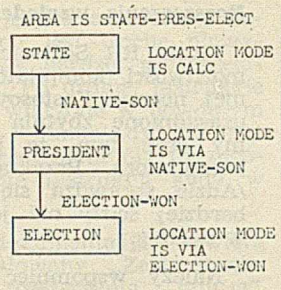
8. Który z wymienionych atrybutów rekordów członkowskich gwarantuje, że dany rekord w trakcie operacji STORE będzie zawsze umieszczany we właściwych wystąpieniach kolekcji?

a) MANDATORY,
b) FIXED,
c) OPTIONAL,
d) AUTOMATIC,
e) MANUAL.

9. Które z wymienionych pojęć DBTG-owskich umożliwiają pełną identyfikację pojedynczego rekordu?

a) obszar cobolowski (ang. realm) lub obszar DDL-owski (ang. area),
b) wskaźniki stanu bieżącego (ang. currency indicators),
c) sposób lokowania (ang. location mode),
d) pozycja logiczna (ang. logical position).

10. Dany jest DBTG-owski schemat struktury danych oraz fragment programu wyszukującego (rysunek i wydruk)



Schemat struktury danych DBTG

```
        MOVE ILLINOIS TO STATE-NAME
        FIND ANY STATE
NEXT-PRES: FIND NEXT PRESIDENT IN NATIVE-
        SON IF DATABASE-STATUS ='DONE'
        GO TO QUIT
NEXT-ELECT: FIND NEXT ELECTION IN ELECTION-
        -WON IF DATABASE-STATUS ='DONE'
        GO TO NEXT-PRES
        IF ELECTION-YEAR ≠'1860'
        GO TO NEXT-ELECT
DONE:   FIND OWNER OF ELECTION-WON
QUIT:   ...
```

Segment programu wyszukującego

Ile zapisów w tabeli wskaźników stanu bieżącego oraz ile różnych wartości tych wskaźników zarejestruje system zarządzania bazą danych w momencie dojścia przez podany program po raz pierwszy od zdania „DONE"?

a) 7 wskaźników, 4 różne wartości,

b) 7 wskaźników, 3 różne wartości,

c) 6 wskaźników, 3 różne wartości,

d) 6 wskaźników, 4 różne wartości.

ROZWIĄZANIA

1 — a, c; 2 — Ia, IIc, IIId, IVb; 3 — b;
4 — a, c, d; 5 — b, d; 6 — c;
7 — a, c; 8 — d; 9 — b, c, d; 10 — b.

LITERATURA I KOMENTARZE

Ad. 1. ACM Computing Surveys. Special Issue: Data-Base Management Systems, Vol. 8, No. 1, pp. 10—12, March 1976. Date C.J.: Wprowadzenie do baz danych. Warszawa WNT, pp. 21—23, 1981.

Ad. 2. ACM Computing Surveys, ibid., pp. 15—28, 30—33. Martin J.: Principles of Data-Base Management. Prentice-Hall, Englewood Cliffs, pp. 148—165, 205—220, 1976.

Ad. 3. ACM Computing Surveys, ibid., pp. 10, 12, 143—146. Martin J.: Organizacja baz danych. Warszawa PWN, pp. 208—265, 426—435, 1983

Ad. 4. ACM Computing Surveys, ibid., pp. 55—56, 147—149. Date C.J.: ibid., pp. 419—423.

Ad. 5. Date C.J.: ibid., pp. 20—21. Martin J.: ibid., pp. 213—216. Na pytania a oraz c system nie może odpowiedzieć ze względu na zaistniałą „pułapkę powiązań" (ang. connection trap problem).

Ad. 6. Date C.J.: ibid., pp. 153—154, 165—168. Tsichritzis D. C., Lochovsky F. H.: Data Base Management Systems. Academic Press, New York, pp. 32—57, 1977.

Ad. 7. CODASYL Data Description Language Committe. Data Description Language Journal of Development. National Bureau of Standards Handbook 113, pp. 2.28—2.29, June 1973. ACM Computing Surveys, ibid., pp. 70—77.

Ad. 8. CODASYL, ibid., pp. 2.30—2.31. ACM Computing Surveys, ibid., pp. 80—82. Date C.J.: ibid., pp. 321—323.

Ad. 9. Tsichritzis D. C., Lochovsky F. H.: ibid., pp. 143—162. ACM Computing Surveys, ibid., pp. 77, 92—98.

Ad. 10. ACM Computing Surveys, ibid., pp. 97—99. Date C.J.: ibid., pp. 336—339.

Adaptowali z angielskiego:

ADAM B. EMPACHER
JERZY J. ROSSOWSKI

[1]) Self-Assessment Procedure V. Part A. General Concepts. Communications of the ACM, Vol. 21, No. 8, pp. 687—693, August 1978. Przedruk na podstawie generalnego zezwolenia stowarzyszenia ACM na wykorzystanie Self-Assessment Procedure V do celów niekomercyjnych.

WARUNKI PRENUMERATY NA 1987 R.

Prenumeratorzy zbiorowi — jednostki gospodarki uspołecznionej, instytucje i organizacje społeczne zamawiają prenumeratę dokonując wpłat na blankiecie „polecenie przelewu".

Prenumeratorzy indywidualni — osoby fizyczne zamawiają prenumeratę dokonując wpłaty w UPT lub NBP na blankiecie Wydawnictwa lub blankiecie NBP. Na odwrocie wszystkich odcinków blankietu należy wpisać tytuł czasopisma, okres prenumeraty, liczbę zamawianych egzemplarzy oraz wartość wpłaty.

Wpłacać należy na konto NBP III O/M Warszawa 1036-7490-139-11.

Prenumerata ulgowa — przysługuje wyłącznie osobom fizycznym — członkom SNT, studentom i uczniom szkół zawodowych. Warunkiem prenumeraty ulgowej jest poświadczenie blankietu wpłaty (przed jej dokonaniem) na wszystkich odcinkach pieczęcią Koła SNT, wyższej uczelni lub szkoły.

Sposób zamawiania prenumeraty taki sam jak dla prenumeraty indywidualnej.

Prenumerata ze zleceniem wysyłki za granicę — zamawia się tak jak prenumeratę indywidualną. Dodatkowo należy podać na blankiecie wpłaty nazwisko i dokładny adres odbiorcy. Cena prenumeraty ze zleceniem wysyłki za granicę jest dwukrotnie wyższa.

Przedpłaty na prenumeratę przyjmowane są w terminach:

— do 10 listopada na I kwartał, I półrocze i cały rok następny,
— do 28 lutego na II, III, IV kwartał i II półrocze,
— do 31 maja na III, IV kwartał i II półrocze,
— do 31 sierpnia na IV kwartał.

Uwaga!

Wpłaty na dwumiesięczniki przyjmowane są na okresy półroczne lub roczne.

Informacji o prenumeracie udziela — Zakład Kolportażu Wydawnictwa NOT-SIGMA, ul. Bartycka 20, 00-716 Warszawa, lub skr. poczt. 1004, 00-950 Warszawa, tel. 40-00-21 w. 249, 293, 297, 299 oraz 40-35-89 i 40-30-86.

Egzemplarze archiwalne czasopism — można nabyć za gotówkę w Klubie Prasy Technicznej w Warszawie ul. Mazowiecka 12, tel. 27-43-65 lub zamówić w Dziale Handlowym Wydawnictwa, ul. Bartycka 20, skr. poczt. 1004, 00-950 Warszawa, tel. 40-37-31, na rachunek dla instytucji lub za zaliczeniem pocztowym dla osób fizycznych.

Cena miesięcznika INFORMATYKA została ustalona na 150 zł za numer (50 zł — cena ulgowa).

Cena prenumeraty wg cennika					
kwartalna		półroczna		roczna	
normalna	ulgowa	normalna	ulgowa	normalna	ulgowa
450	150	900	300	1800	600

Janus/Ada — podzbiór Ady dla mikrokomputerów

Janus/Ada firmy RR Software jest realizacją niestandardowego podzbioru Ady. Jest ona przeznaczona do pracy w systemach operacyjnych CP/M-80, CP/M-86 albo MS-DOS (PC-DOS), na komputerach, które dysponują przynajmniej 56 KB pamięci i dwoma stacjami dysków elastycznych. Na pełny produkt składają się: kompilator, program łączący, asembler i disasembler.

Kompilator zapewnia wykrywanie błędów kompilacji i wykonania, bardzo przydatne na etapie uruchamiania programów. Dokumentacja Janus/Ady jest przeznaczona do wykorzystania łącznie z podręcznikiem języka.

Omówienie Janus/Ady

Podzbiór Ady zrealizowany przez RR Software ma wszystkie konstrukcje języka niezbędne do poważnej pracy aplikacyjnej. Wśród dostępnych typów danych znajdują się znakowe, dyskretne, bajtowe (typ niestandardowy), wyliczeniowe, logiczne, tablicowe (ograniczone do jednego wymiaru), rekordowe (z wariantami, ale bez wyróżników), napisowe (realizacja niestandardowa) i typy definiowane przez użytkownika, podtypy oraz typy pochodne. W 16-bitowej wersji Janus/Ady są także liczby rzeczywiste i przeciążanie; niedługo te cechy będzie miała wersja 8-bitowa. Kompilator stawia do dyspozycji pełny zbiór operatorów arytmetycznych, logicznych, relacji i przynależności. Zgodnie z normą Ady możliwa jest też pewnego rodzaju jawna konwersja typów.

Janus/Ada wykonuje wszystkie najbardziej użyteczne instrukcje języka Ada, m.in. instrukcje warunkowe **IF** i **CASE**, przypisanie, skoki **GOTO**, **EXIT** i **RETURN** oraz pętle **LOOP**, **FOR** i **WHILE**. Co najważniejsze, Janus/Ada dopuszcza rozłączną kompilację pakietów (modułów), także z typami prywatnymi. Można używać w pełni zgodnych z normą podprogramów.

Janus/Ada a norma Ady

Pełna Ada zapewnia możliwości, których jeszcze nie ma w Janus/Adzie. Są to: współbieżność (przetwarzanie równoległe), obsługa wyjątków (błędów), rodzajowość (wzorce pakietów i procedur czasu kompilacji) i specyfikację reprezentacji (pamięci fizycznej). Poza tymi wyjątkami Janus/Ada jest zasadniczo zgodna z obecną normą Ady. Choć kompilator odbiega od standardowej Ady jeszcze w wielu innych miejscach, nie powinno to prowadzić do większych kłopotów. Warto podkreślić, że Janus/Ada ma pragmę **WARNING** (tabela 1), która powoduje wytwarzanie przez kompilator komunikatów ostrzegawczych dla wszystkich konstrukcji niestandardowych.

W tabeli 1 wymieniono bardziej szczegółowo inne pragmy Janus/Ady (pragmy są dyrektywami kompilatora, które sterują procesem kompilacji). Podstawowy opis wskazuje, jak przydatne mogą być pragmy, szczególnie przy uruchamianiu programu.

W tabeli 2 przedstawiono atrybuty zrealizowane w Janus/Adzie. Atrybuty są właściwościami typów danych. Opis tych atrybutów może niewiele mówić osobom nie znającym dobrze Ady.

Zarówno pragmy jak i atrybuty Janus/Ady stanowią niestandardowy podzbiór tego, co w Adzie powinno się znajdować.

Rozszerzenia względem normy

Firma RR Software nie ograniczyła możliwości kompilatora tylko do samej normy. Zastosowano rozszerzenia uzasadnione zbytnią sztywnością normy oraz tym, że polepszały jakość kompilatora. Przykładowo, w Janus/Adzie sprawdza się typy w sposób bardziej ścisły niż jest to wymagane w normie.

Należy wspomnieć o dwóch głównych rozszerzeniach. Pierwszym jest warunkowa kompilacja. Linie, które mają być kompilowane warunkowo, są poprzedzone znakiem „**%**". Proces kompilacji jest wtedy sterowany przez polecenia do kompilatora (jeśli wszystkie linie warunkowe należy skompilować) lub przez pragmę **CONDCOMP** (tabela 1). Jest to cecha bardzo przydatna przy uruchamianiu programów.

Bardzo przydatne jest drugie odstępstwo od normy — kompilacja kodu asemblerowego „w linii". Instrukcja ASM w Janus/Adzie jest trochę inna niż instrukcja kodu w Adzie, choć na niej oparta. Po pierwsze, instrukcje ASM używają argumentów całkowitych, co pociąga konieczność operowania dwójkowymi kodami rozkazów lub stałymi odpowiadającymi mnemoni-

Tabela 1. Pragmy Janus/Ady

Nazwa	Opis
ARITCHECK	Kontrola przepełnień arytmetycznych (On/Off)
CONDCOMP	Kompilacja warunkowa (Off/On); ustawiana również poleceniem do kompilatora
DEBUG	Tworzenie informacji zwrotnej o błędach wykonania (On/Off); ustawiane również poleceniem do kompilatora
ENUMTAB	Prowadzenie wykazów do wprowadzania i wyprowadzania mnemoników wartości wyliczeniowych (On/Off)
INCLUDE	Przyjmuje jako argument napis będący nazwą pliku, który ma być włączony „w linii" do programu źródłowego; tej pragmy nie można zagnieżdżać
LIST	Sterowanie wydrukiem programu źródłowego tworzonym przez kompilator (On/Off); jeżeli odpowiednia opcja list jest wymieniona w poleceniu do kompilatora
RANGECHECK	Sprawdzanie zakresu dla zmiennych i indeksów tablic (On/Off)
VERBOSE	Sterowanie długością komunikatów o błędach (On/Off); ustawienie (On) powoduje wyprowadzanie dwóch linii poprzedzających błąd, ze strzałką wskazującą błędną składnię; skasowanie (Off) powoduje wyprowadzenie tylko numeru linii; ustawiane również poleceniem do kompilatora
WARNING	Ostrzeganie o użyciu niestandardowych konstrukcji Ady

Tabela 2. Atrybuty Janus/Ady

Nazwa	Opis
X'Address	Adres początkowy (typu integer) dowolnego obiektu klub podprogramu
T'Size	Liczba bitów (typu integer) potrzebna typowi lub podtypowi T
T'First	Dolne ograniczenie lub wartość minimalna dla dyskretnego typu lub podtypu T (wynik jest również typu T)
T'Last	Górne ograniczenie lub wartość maksymalna dla dyskretnego typu lub podtypu T (wynik jest również typu T)
T'Pos(X)	Położenie (typu integer) wartości X w ciągu wartości dyskretnego typu lub podtypu T
T'Val(X)	Udostępnienie wartości typu T na pozycji X; X musi być typu integer, a T musi być typem lub podtypem dyskretnym
T'Pred(X)	Udostępnienie poprzednika wartości X, typu T, który musi być podtypem lub typem dyskretnym
T'Succ(X)	Udostępnienie następnika wartości X, typu T, który musi być typem lub podtypem dyskretnym
P'Location	Udostępnienie wartości wskaźnika aktualnego adresu (ang. current location counter) dla kodu źródłowego pakietu P

kom instrukcji języka asemblera. Zastosowana metoda jest trochę niewygodna, ale nie tak bardzo, jak mogłoby się wydawać. Po drugie, instrukcje ASM mogą być włączone w dowolnym miejscu pakietu Janus/Ady, co jest mniejszym ograniczeniem niż narzucone na stosowanie instrukcji kodu w Adzie.

Dostarczona dokumentacja zawiera dodatek zatytułowany „Zastrzeżenia, ograniczenia i rozszerzenia Janusa". Omówiono w nim rozbieżności między Janus/Adą a pełną normą Ady. Niewielka długość dodatku (trzy i pół strony) wskazuje, że ważnych rozbieżności powinno być niewiele.

Użyteczność bibliotek jest często wskazówką możliwości danej realizacji języka. Firma RR Software załącza zbiór bardzo przydatnych bibliotek określonych przez normę.

Do zwyczajnego wprowadzania i wyprowadzania danych służy zbiór procedur dla plików sekwencyjnych i tekstowego wejścia-wyjścia. Realizacja bibliotek niewiele odbiega od normy. Wśród procedur operujących plikami istnieją: **Create, Open, Read, Write, Close, Delete i End—of—File**, a w bibliotece tekstowego wejścia-wyjścia: **Get, Put, Get—Line** i odpowiednie odwołania w celu zmiany przyporządkowania standardowych plików wejściowych i wyjściowych.

Standardowa Ada dopuszcza biblioteki definiowane przez dostawcę lub użytkownika (w rzeczywistości ich wymaga). Janus/Ada jest dostarczana z kilkoma ciekawymi bibliotekami zdefiniowanymi przez producenta. Dołączony jest zbiór procedur do zarządzania stogiem (pamięcią obszaru roboczego) i do bitowych oraz bajtowych operacji na liczbach całkowitych. Istnieje biblioteka programowego sterowania łańcuchami programów i wykonywaniem nakładek (niektóre procedury są niedostępne w systemach ośmiobitowych). Istnieje również grupa rozmaitych procedur wejścia-wyjścia wraz z logicznym wprowadzaniem i wyprowadzaniem zawartości sektora w operacjach blokowych na dysku. Oprócz tego, istnieje duży zbiór elementarnych operacji napisowych, zawierający m.in. operacje konwersji typów.

Problem ośmiu hetmanów

Na wydrukach 1 i 2 przedstawiono napisane w Adzie rozwiązanie problemu ośmiu hetmanów. Oryginalny program pochodzi z książki: Narain Gehani, „Ada, an advanced introduction", (Prentice-Hall, Englewood Cliffs (NJ), 1983). Zadanie polega na rozmieszczeniu na szachownicy ośmiu hetmanów tak, aby nie atakowały się wzajemnie. Dokładniejsze objaśnienie algorytmu rozwiązania znajduje się w książce Gehaniego.

W Janus/Adzie ten program składa się z dwu pakietów: **ATEQENS** i **CHESSBD.** Pierwszy jest pakietem głównym, do którego **CHESSBD** jest dołączany instrukcją WITH. Pakiet **ATEQENS** jest pełnym pakietem, a nie procedurą, ponieważ w Janus/Adzie wszystkie moduły kompilowane oddzielnie muszą być pakietami. Wszy-

```
--
-- Pakiet do dzialan na szachownicy. Pakiet zawiera nastepujace
-- procedury:
--
--   PUT_QUEEN - umieszcza hetmana na polu: wiersz, kolumna.
--
--   REMOVE_QUEEN - zdejmuje hetmana z szachownicy.
--
--   PRINT_POSITION - drukuje aktualny uklad figur na szachownicy.
--
--
package CHESSBD is

   procedure PUT_QUEEN (ROW,COL: in INTEGER);

   procedure REMOVE_QUEEN (ROW,COL: in INTEGER);

   function SAFE (ROW,COL: INTEGER) return BOOLEAN;

   procedure PRINT_POSITIONS;

end CHESSBD;

--
-- CHESSBD korzysta z nastepujacych struktur danych:
--
--   ROW_POS (I) - rzad okreslajacy pozycje hetmana I w kolumnie I.
--
--   R(J) - prawda, jesli w rzedzie J nie ma hetmana, inaczej falsz.
--
--   LD(K) - prawda, jesli na K-tej przekatnej skierowanej w lewo
--           w dol, nie ma hetmana.
--
--   RD(K) - prawda, jesli na K-tej przekatnej skierowanej w prawo
--           w dol, nie ma hetmana.
--
--
-- tresc pakietu do dzialan na szachownicy
--
package body CHESSBD is

   ROW_POS: array (1..8) of INTEGER;
   R: array (1..8) of BOOLEAN;
   LD: array (..16) of BOOLEAN;
   procedure PUT_QUEEN (ROW,COL: integer) is
--
--umiesc hetmana na szachownicy, na polu ROW, COL
--
   begin
      ROW_POS (COL) := ROW;
      R (ROW) := FALSE;
      LD (ROW+COL) := FALSE;
      RD (COL-ROW) := FALSE;
   end PUT_QUEEN;

   procedure REMOVE_QUEEN (ROW, COL: in INTEGER) is
--
-- zdejmij hetmana z pola szachownicy: ROW, COL .
--
   begin
      R (ROW) := TRUE;
      LD (COL+ROW) := TRUE;
      RD (COL-ROW) := TRUE;
   end REMOVE_QUEEN;

   function SAFE (ROW, COL: in INTEGER) return BOOLEAN is
--
-- okresl czy hetman bylby bezpieczny (SAFE) na polu ROW, COL.
--
   begin
      return R (ROW) and LD (COL + ROW) and  RD (COL - ROW);
   end SAFE;
   procedure PRINT_POSITIONS is
--
-- podaj aktualny uklad figur na szachownicy
--
   begin
      for I in 1..8 loop
         PUT ("queen number "); PUT (I);
         PUT (" is in column "); PUT (I);
         PUT (" and row "); PUT (ROW_POS (I));
         NEW_LINE;
      end loop;
   end PRINT_POSITIONS;

begin
--
--    inicjalizuj rozmaite struktury danych
--
   for I in 1..8 loop R (I) := TRUE; end loop;
   for I in 2..16 loop LD (I) := TRUE; end loop;
   for I in -7..7 loop RD (I) := TRUE; end loop;

end CHESSBD;
```

Wydruk 1. Pakiet CHESSBD napisany w Janus/Ada

```
with TEXT_IO; use TEXT_IO;
procedure EIGHT_QUEENS is
   package IO_INTEGER is new INTEGER_IO(INTEGER);
   use IO_INTEGER;
   --wstaw tutaj opis i tresc pakietu CHESS_BOARD
   use CHESS_BOARD;

   procedure PLACE_QUEENS(I: in INTEGER;
                          SUCCESSFUL: out BOOLEAN) is
      J: INTEGER := 0;
         --J jest numerem wiersza, I jest numerem kolumny
   begin
      SUCCESSFUL := FALSE;
      while not SUCCESSFUL and J /= 8 loop
         J := J+1;
         if SAFE(J,I) then
            PUT_QUEEN(J,I);
            if I < 8 then
               PLACE_QUEENS(I+1,SUCCESSFUL);
               if not SUCCESSFUL then
                  REMOVE_QUEEN(J,I);
               end if;
            else
               SUCCESSFUL := TRUE;
            end if;
         end if;
      end loop;
   end PLACE_QUEENS;

   SUCCESSFUL: BOOLEAN;
begin  --tresc glownej procedury EIGHT_QUEENS
   PLACE_QUEENS(1,SUCCESSFUL);  --ustaw wszystkie 8 figur
   if SUCCESSFUL then
      PRINT_POSITIONS;
         --w rzeczywistosci nie potrzeba sprawdzac ze procedura
         --PLACE_QUEENS daje dobry wynik, poniewaz wiadomo ze
         --poszukiwane ustawienia istnieja.
   end if;
end EIGHT_QUEENS;
```

Wydruk 2. Pakiet EIGHT_QUEENS wg podręcznika Gehaniego

stkie pakiety, które składają się na pełny program, muszą mieć specyfikacje i ciało, za wyjątkiem pakietu głównego (podstawowego). Jeżeli pakiet nie ma specyfikacji, to jest uważany za pakiet główny.

Należy zauważyć, że struktury danych w ciele pakietu **CHESSBD** nie są inicjowane gdy się je deklaruje. Obecnie Janus/Ada wyklucza inicjalizację tablic w czasie deklarowania (mimo że dla większości obiektów dyskretnych jest inaczej). W konsekwencji, na końcu ciała pakietu znajduje się część inicjalizacyjna. Ta część programu jest wykonywana jeden raz, kiedy pakiet zostaje wywołany (załadowany).

Należy też podkreślić, że w Janus/Adzie nie ma bibliotek jawnie dołączanych instrukcją WITH ani USE. Kompilator Janus/Ada włącza jawnie pewne podstawowe elementy biblioteki, jako części składni wywołania. Nie są one traktowane jako jednostki odrębne. Jest to przykład niewielkiego odejścia od zalecanych zasad w celu wykonania kompilatora łatwiejszego w użyciu. Ponadto, nazwy pakietów muszą być takie same jak ich plików źródłowych.

* * *

Artykuł ten jest pomyślany jako krótkie wprowadzenie do kompilatora Janus/Ady, wykonanego i sprzedawanego przez firmę RR Software. Pomimo że kompilator nie jest w pełni zgodny ze standardową Adą, jest przydatny do nauczania tego języka oraz do rozwijania poważnych zastosowań i zapewnia względnie łatwe przejście do używania kompilatora pełnej Ady na mikrokomputery, kiedy taki będzie dostępny.

Kompilator Janus/Ady jest napisany całkowicie w Janus/Adzie, co jest oznaką wysokiej jakości tego programu. Wprowadzono go do sprzedaży na początku 1982 roku i do tej pory ma już dużą grupę użytkowników.

Oprac.: Jerzy STRYCHARCZYK
na podst. J. of Pascal and Ada, Vol. 3, No. 4, 1984

Kostki z arsenku galu

Technika krzemowa natrafia na ograniczenia związane przede wszystkim z szybkością działania urządzeń. Jednym z rozwiązań tego problemu jest wykorzystanie arsenku galu jako materiału zastępującego krzem. Jako jedna z pierwszych oferuje swe usługi w tym zakresie firma Gigabit Logic z Newbury Park w stanie Kalifornia (USA), która opracowała proces wytwarzania takich kostek zwany D-MESFET pozwalający na uzyskiwanie układów o rozstępie ścieżek 1 μm i opóźnieniu 100 ps na tranzystor. W zbudowanych za 19 mln dolarów zakładach znajdujących się również w Newbury Park, firma Rockwell International Corp. wytwarza dla Defense Advanced Research Projects Agency pierwsze kostki pamięci operacyjnej o pojemności 1 Kb. Linia pilotowa może wytwarzać 250 3-calowych płytek tygodniowo. Po usprawnieniach liczba wytwarzanych kostek zwiększy się cztero- a nawet szesnastokrotnie. Jest przewidziane opracowanie matryc programowanych zawierających sześć tysięcy tranzystorów.

Specjalizująca się w produkcji superkomputerów firma Cray Research Inc. podała, że opracowano już kilkanaście pierwszych układów GaAs spośród około stu, jakie mają być zastosowane w systemie Cray-3. Zawierają one po 300—500 tranzystorów polowych o złączach metal-półprzewodnik. Są one wytwarzane w pilotowych zakładach firmy w Chippewa Falls w stanie Wisconsin. Cray-3, którego dostawy są przewidziane w końcu 1987 roku, składać się będzie z 16 procesorów i będzie dziesięciokrotnie szybszy od komputera Cray-2 wykonującego 1,2 mld operacji zmiennoprzecinkowych na sekundę.

Natomiast naukowcy z uniwersytetu stanu Illinois opracowali proces nakładania cienkich warstw GaAs na płytkę krzemową, przy czym dla uzyskania jak najmniejszej liczby defektów płytka ta jest przecięta wzdłuż płaszczyzny nachylonej pod kątem 4° w celu dopasowania geometrycznego siatek krystalicznych krzemu i arsenku galu. Dodatkowe zmniejszenie tych defektów osiąga się przez nałożenie na warstwy arsenku związku InGaAs. Dzięki trzykrotnie większej przewodności cieplnej krzemu, układy GaAs utworzone na takim podłożu mogą być bardziej zwarte i mogą pracować z większą mocą. Uzyskuje się też obniżkę kosztów, gdyż warstwy GaAs mogą być nakładane na większych powierzchniach aniżeli standardowe płytki 3-calowe. W przyszłości może to doprowadzić do opracowania układów nowego rodzaju, które będą wykorzystywać zarówno właściwości optoelektroniczne, jak i szybkie przełączenie GaAs, a także większą gęstość krzemu. Dotychczas udało się opracować wielozłączowe tranzystory polowe pracujące na częstotliwościach do 20 GHz. Trwają prace nad laserem półprzewodnikowym o pracy ciągłej opartym na tych układach.

Japońska firma NEC w końcu lutego 1986 roku oferowała pierwsze trzy kostki GaAs kompatybilne z układami ECL, obiecując w niedługim czasie ofertę znacznie rozszerzyć. Wykorzystują one możliwości arsenku galu, jeśli chodzi o szybkość, przy napięciach zasilających takich, jakie są stosowane w układach ECL. Cena tych układów wynosiła 253 dolary, tj. około dwa razy więcej niż układów ECL. Jedna z kostek ma trójwejściowe bramki OR/NIL, a dwie pozostałe przerzutniki typu D i typu T, które pracują z zegarem o częstotliwości 2 GHz, a więc dwukrotnie wyższej niż dla układów krzemowych. Czasy propagacji dla bramek wynoszą około 250 ps, a dla przerzutników 400 ps. Czas narastania impulsów dla wszystkich tych układów wynosi około 130 ps, a opadania 120 ps. Pobór prądu waha się od 50 mA dla bramki do 70 mA dla przerzutnika typu D.

Łączną wartość układów GaAs, które będą wyprodukowane w 1986 roku, ocenia się na 240 mln dolarów. Przewiduje się wzrost tej wartości do 5 mld w ciągu następnego dziesięciolecia, przy czym w latach dziewięćdziesiątych dominować będą układy analogowe. Zwiększy się też (do 79%) udział układów monolitycznych w stosunku do hybrydowych, które obecnie stanowią większość układów GaAs.

Najnowszą ofertą w tej dziedzinie jest standardowa rodzina układów Q-Logic, o częstotliwości zegara 3 GHz, firmy TriQuint Semiconductor, 15 razy szybsza od krzemowych układów CMOS i pięć razy szybsza od ECL. Układy te nadają się m.in. do systemów łączności na światłowodach. Mogą one pracować poprawnie przy napięciu 4—5,2 V i temperaturze 0—+85°C (opracowywana jest wersja na zakres —55..+125°C). Badania niezawodnościowe wykazały średni czas pracy bezawaryjnej równy 10 milionom godzin przy temperaturze 125°C. Wśród tych układów są liczniki, układy dzielące i multipleksery. Można je nabywać w standardowym płaskim wykonaniu, a także w wersji ceramicznej, wielowarstwowej do bardzo szybkich zastosowań. Cena waha się w granicach od kilkudziesięciu do dwustu kilkudziesięciu dolarów; czas dostawy 45 dni.

Opracował: JAN RYŻKO
na podst. Electronics nr 5, 6 i 9/1986

Kto jest kim w IFIP

Blagovest H. Sendov

Akademik Blagovest H. Sendov — wiceprezydent IFIP — urodził się w 1932 r. w Assenovgradzie, w południowej Bułgarii. Studiował matematykę na Uniwersytecie Sofijskim, gdzie w 1964 r. otrzymał tytuł doktora filozofii. Tytuł doktora nauk przyrodniczych uzyskał w 1967 r. w Instytucie Matematycznym Akademii Nauk ZSRR (Moskwa). W 1963 r. został mianowany docentem, a obecnie jest już profesorem nauk komputerowych. Był przez trzy lata dziekanem Wydziału Matematyki oraz przez sześć lat rektorem Uniwersytetu Sofijskiego. Przewodniczy Sekcji Modeli Matematycznych w Centrum Matematyki i Mechaniki Bułgarskiej Akademii Nauk i Uniwersytetu Sofijskiego, a także Sekcji Badań Operacyjnych w tym samym Centrum. Jest wiceprzewodniczącym i generalnym sekretarzem naukowym Bułgarskiej Akademii Nauk, a także, od 1976 r. członkiem bułgarskiego parlamentu.

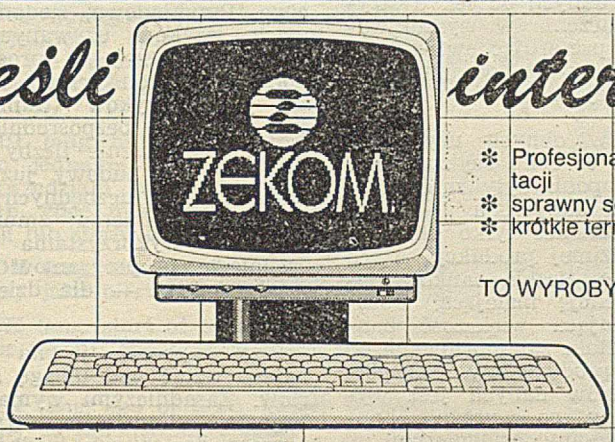
Blagovest Sendov uczestniczy w pracach IFIP od 1965 r., kiedy został wybrany bułgarskim przedstawicielem do Komitetu Technicznego Opieki Zdrowotnej i Badań Biologicznych. Jest także aktywnym działaczem Komitetu Technicznego ds. Oddziaływań między Komputerami a Społeczeństwem. W 1973 r. otrzymał Srebrny Medal IFIP-u. W 1983 r. został wybrany do zarządu IFIP, a jego wiceprezydentem został w 1984 r.

Zainteresowania zawodowe B. Sendova obejmują teorię aproksymacji i modelowanie matematyczne w biologii. Na te tematy opublikował wiele artykułów naukowych. W 1969 r. otrzymał Nagrodę Naukową im. Dymitrowa.

Blagovest Sendov należy do wielu organizacji zawodowych, między innymi do Związku Matematyków Bułgarskich i Amerykańskiego Towarzystwa Matematycznego. Od 1980 r. jest wiceprzewodniczącym Międzynarodowego Stowarzyszenia Uniwersytetów, a od 1981 r. wiceprezydentem Światowej Rady Pokoju. Obecnie zajmuje się badaniem szacowania błędu w analizie numerycznej. W marcu 1986 r., podczas posiedzenia Rady IFIP w Monachium przedstawił ostatnie prace nt. badań „stałych Whitney'a". Problem ten czekał na rozwiązanie prawie 30 lat.

MK

Jeśli interesuje Państwa



* Profesjonalny sprzęt o wysokiej jakości, niezawodny w eksploatacji
* sprawny serwis
* krótkie terminy dostaw, lub dostawy natychmiastowe

TO WYROBY ZEKOMU SĄ DO PAŃSTWA DYSPOZYCJI

Zakład Elektroniki Komputerowej

oferuje terminale ekranowe:

MV 2580 Standard VT 52 firmy DEC / odpowiednik MERA 7953. Przeznaczony do pracy w systemach komputerowych wyposażonych w kanał transmisji V 24 lub pętli prądowej 20/60 mA – jako końcówka zdalnego dostępu.

MV 2581 Odpowiednik MERA 7911 N.
Przeznaczony do pracy w systemach komputerowych ODRA 1300 wyposażonych w jednostkę sterującą MERA 7802.

MV 2582E Odpowiednik terminala typu 7181/2 firmy ICL.
Przeznaczony do pracy w systemach komputerowych ODRA 1300, ICL 1900, ICL 2900, ICL system 4.

MR 1240 Odpowiednik MERA 7951.
Przeznaczony do wprowadzania danych do systemu MERA 9150 lub systemu REDIFON.

ZAKŁAD ELEKTRONIKI KOMPUTEROWEJ ul. Makowa 8, 91-480 Łódź tel. 34 30 49

EO/307/87

Stan usług w dziedzinie banków danych w Japonii

Japonia kojarzy się powszechnie z przodującą techniką i wysokim stopniem rozwoju gospodarczego. Są jednak dziedziny, z rozwoju których Japończycy nie są zadowoleni. Jak wynika z przedstawionego w listopadzie 1985 r. raportu japońskiego Podkomitetu ds. Promocji Dostarczania Usług Informacyjnych, do nich należą usługi w dziedzinie banków danych. Wspomniany raport oraz inne materiały ilustrujące aktualny stan, problemy i perspektywy usług informacyjnych w Japonii opublikowano w numerze 65 kwartalnika Japan Computer Quarterly (kwiecień 1986 r.), który w całości został poświęcony tej tematyce. Poniżej przedstawiono niektóre charakterystyczne opinie w nim zawarte.

Japonia późno zdała sobie sprawę z potrzeb w dziedzinie banków danych. Powołanie w kwietniu 1986 r. wspomnianego podkomitetu, działającego w ramach Komitetu ds. Przemysłu Informatycznego, jest wyrazem zdecydowania Japonii, aby tworzeniu banków danych i promocji ich usług nadać wysoki priorytet. W raporcie stwierdza się, że Japończycy, jako pojedyncze jednostki, są mocno ograniczeni jeśli chodzi o ilość i rodzaj informacji, które mogą uzyskać spośród zalewających ich ogromnych masywów danych. Jednocześnie wskazuje się, że to właśnie banki danych mogą umożliwić sprawne i skuteczne wykorzystanie informacji, zwiększyć stopień jej wykorzystania oraz pozwalają uniknąć dublowania wysiłków, czasu i pieniędzy przy zbieraniu potrzebnych danych. Banki danych muszą być traktowane jako życiowo ważne zasoby społeczne, które razem ze sprzętem, oprogramowaniem i zasobami ludzkimi tworzą jeden z filarów podtrzymujących zaawansowane społeczeństwo informacyjne.

Dochody uzyskane w Japonii z usług informacyjnych (produkcja banków danych, ich dystrybucja, pośredniczenie przy wyszukiwaniu informacji i in.) w 1984 r. wyniosły 96,7 mld jenów (ok. 402,9 mln dol. USA), tj. ok. $^1/_5$ tego rodzaju dochodów zarejestrowanych w tym samym czasie w Stanach Zjednoczonych i $^1/_3$ zarejestrowanych w Europie. Osiągane w ostatnich latach przez Japonię tempo wzrostu w tej dziedzinie nie wróży szybkiego nadrobienia dystansu. Tempo to było co prawda wysokie, podobnie jak w USA i Europie. Na przykład w 1984 r. wyniosło 18,6%.

W Japonii jest mniej niż 100 firm przejawiających aktywność w dziedzinie banków danych. Większość z nich jest jednak zaangażowana przede wszystkim w działalność innego rodzaju, a usługi informatyczne mają dla nich znaczenie raczej marginesowe i sprowadzają się do udostępniania użytkownikom swoich własnych, prywatnych danych i posiadanych zasobów komputerowych. W przeciwieństwie do sytuacji jaka istnieje w USA i Europie, w Japonii jest bardzo mało dużych przedsiębiorstw specjalizujących się tylko w usługach informacyjnych. Ponadto, inaczej niż się to praktykuje w USA i Europie, japońscy sprzedawcy usług w dziedzinie banków danych nie odróżniają produkcji banków danych od ich dystrybucji i pośredniczenia w wyszukiwaniu informacji, preferując świadczenie usług kompleksowych, obejmujących wszystkie operacje razem.

Liczba banków danych użytkowanych w Japonii co roku się zwiększa. W 1982 r. w Japonii były zarejestrowane 604 banki danych, a w 1984 r. — już 1242. Jednakże tylko ok. 20% z nich jest rodzimej produkcji, a ich udział w ogólnej liczbie banków danych ma tendencję malejącą. Pozostała przygniatająca większość banków danych została zaimportowana od zagranicznych producentów. Eksport japońskich banków danych jest wyjątkowo skromny, co wyraźnie kontrastuje z sukcesami osiągniętymi w dziedzinie rozwoju zaawansowanych technologii oraz z wielkością masywów i atrakcyjnością posiadanych przez Japonię informacji. Fakt, że w tej sytuacji Japonia jest importerem, a nie eksporterem informacji, może być powodem słusznej krytyki ze strony innych państw. Japończycy dostrzegają możliwość takiej krytyki i próbują znaleźć usprawiedliwienie. Zarzut świadomego kształtowania nadwyżki w imporcie usług informacyjnych odpierają wskazując na swe opóźnienie w dziedzinie zbierania informacji i oragnizowania ich w formie banków danych. Z tego wniosek, że opóźnienie może się do czegoś przydać.

WIKTOR RZECZKOWSKI

Tłumaczenie maszynowe

Europejska Komisja Ekonomiczna EWG przyznała holenderskiej firmie BSO sumę 4,5 mln dol. na budowę systemu tłumaczenia maszynowego z wykorzystaniem esperanto jako standardowego języka pośredniego.

System, o roboczej nazwie DLT, ma tłumaczyc dany język źródłowy na esperanto, wykorzystując procesor lokalny. Wygenerowany tekst pośredni może być następnie przesłany do wybranego procesora, który dokona ostatecznego tłumaczenia na konkretny język docelowy. W przeciwieństwie do dotychczasowych prób budowy takich systemów, DLT nie będzie oparty na centralnym komputerze o dużej mocy obliczeniowej, lecz na sieci mniejszych procesorów.

W opinii fachowców wykorzystanie standardowego języka pośredniego w tłumaczeniu maszynowym jest właściwe. Przykładowo, siedem podstawowych języków używanych w EWG wymaga budowy 42 programów dokonujących konwersacji między każdą parą języków (tłumaczenie nie jest procesem bezpośrednio odwracalnym). Zwiększenie liczby języków o trzy wymaga budowy już 90 programów. Liczba niezbędnych programów ulega radykalnemu zmniejszeniu w wypadku wykorzystania języka pośredniego: 14 programów dla siedmiu języków i 20 — dla dziesięciu języków.

Przy wyborze języka pośredniego należy wziąć pod uwagę sześć kryteriów. Zasadniczymi wymaganiami są: bogactwo słownictwa, wyrażalność za pomocą dostępnych środków informatyki, możliwość precyzyjnego zdefiniowania leksemów języka i regularność gramatyki. Pożądanymi cechami są również: podobieństwo do języków naturalnych i istnienie gotowego zasobu wiedzy zgromadzonej w danym języku.

Żaden język naturalny nie spełnia wszystkich, a nawet najważniejszych z wymienionych wymagań; języki sztucznie konstruowane przez informatyków nie spełniają ostatnich dwóch wymagań (a czasem nawet pierwszego z nich). Wydaje się, że w obecnej chwili esperanto jest najbliższy spełnienia tych wymagań.

M. MACHURA
Systems International, sierpień 1986

Polskie nazewnictwo znaków kodu ASCII/ISO (2)

Po obszernym omówieniu terminologii lokalnych sieci komputerowych, wypada powrócić do rozpoczętych w numerze 9—10/1986 rozważań dotyczących polskiego nazewnictwa znaków kodu ASCII. Ważną grupę znaków tego kodu stanowią, nie mające reprezentacji graficznej, tzw. znaki sterujące. Odpowiadające im ciągi kodowe mieszczą się prawie całkowicie w zakresie 0—31 (00H—1FH w zapisie szesnastkowym), a ich nazwy międzynarodowe są skrótami odpowiednich określeń w języku angielskim.

Polskojęzyczne nazwy tych znaków określono w normie PN-78/T-42108. Poniżej zweryfikowano te nazwy, przyjmując następujące kryteria:

- powinny mieć formę rzeczownikową,
- muszą dokładnie wyrażać znaczenie znaku (określone we wspomnianej normie i w normie ISO 2382/IV),
- powinny być zgodne z systemem ustalonych terminów informatycznych.

Choć zarówno w Polskiej Normie, jak i w normie ISO znaki sterujące pogrupowano w odpowiednie kategorie (PN: znaki sterowania transmisją, znaki sterowania formatem, znaki sterowania urządzeniami, separatory informacji, znaki rozszerzenia i znaki nie mające ogólnej nazwy; odpowiednich kategorii ISO jest nieco mniej), wydaje się, że bardziej przejrzyste będzie omówienie ich w kolejności zgodnej z występującą w kodzie ASCII. Warto też dodać, że wyjaśnienia opisowe występujące przy nazwie każdego znaku nie pretendują do roli definicji, a stanowią raczej moją interpretację określeń z obu wspomnianych norm.

0 (00H) — **NUL** (ang. null), **brak informacji.**
Znak, którego jedynym przeznaczeniem jest wypełnienie nośnika lub przedziału czasowego bez wpływu na zawartość informacji, choć może mieć uboczny wpływ na format informacji lub sterowanie urządzeniem.

1 (01H) — **SOH** (ang. start of header), **początek nagłówka.**
Znak stosowany jako pierwszy w nagłówku wiadomości.

2 (02H) — **STX** (ang. start of text), **początek tekstu.**
Pierwszy znak tekstu, oznaczający także koniec nagłówka wiadomości.

3 (03H) — **ETX** (ang. end of text), **koniec tekstu.**
Znak stosowany do zakończenia tekstu.

4 (04H) — **EOT** (ang. end of transmission), **koniec transmisji.**
Znak służący do zakończenia transmisji jednego lub kilku tekstów i odpowiednich nagłówków.

5 (05H) — **ENQ** (ang. enquiry), **zapytanie.**
Znak służący do żądania odpowiedzi od stacji, z którą nawiązano połączenie.

6 (06H) — **ACK** (ang. acknowledge), **potwierdzenie.**
Znak oznaczający odpowiedź pozytywną.

7 (07H) — **BEL** (ang. bell), **sygnał dźwiękowy.**
Znak służący do sygnalizacji akustycznej.

8 (08H) — **BS** (ang. backspace), **cofnięcie.**
Znak służący do przemieszczenia kursora w wierszu o jedną pozycję znakową do tyłu.

9 (09H) — **HT** (ang. horizontal tabulation), **tabulacja pozioma.**
Znak służący do przemieszczania kursora w wierszu na określoną pozycję do przodu.

10 (0AH) — **LF** (ang. line feed), **wysuw wiersza.**
Znak służący do przemieszczania kursora na tę samą pozycję w następnym wierszu.

11 (0BH) — **VT** (ang. vertical tabulation), **tabulacja pionowa.**
Znak służący do przemieszczania kursora na tę samą pozycję w jednym z następnych, z góry określonych wierszy.

12 (0CH) — **FF** (ang. form feed), **wysuw strony.**
Znak służący do przemieszczenia kursora na tę samą pozycję następnej strony.

13 (0DH) — **CR** (ang. carriage return), **powrót karetki.**
Znak służący do przemieszczenia kursora na początek tego samego wiersza.

14 (0EH) — **SO** (ang. shift out), **zmiana kodu.**
Znak służący do rozszerzania zestawu znaków graficznych.

15 (0FH) — **SI** (ang. shift in), **powrotna zmiana kodu.**
Znak służący do ponownego zawężenia zestawu znaków graficznych rozszerzonego użyciem znaku SO.

16 (10H) — **DLE** (ang. data link escape), **zmiana znaczenia.**
Znak służący do zmiany znaczenia określonej liczby kolejnych, przyległych znaków, używany wyłącznie w celu rozszerzenia zestawu znaków sterowania transmisją.

17—20 (11H—14H) — **DC1—DC4** (ang. device control), **sterowanie urządzeniem.**
Znaki służące do realizowania różnych funkcji sterowania urządzeniami.

21 (15 H) — **NAK** (ang. negative acknowledge), **odrzucenie.**
Znak oznaczający odpowiedź negatywną.

22 (16H) — **SYN** (ang. synchronous idle), **synchronizacja spoczynkowa.**
Znak służący do utrzymania synchronizacji w stanie spoczynkowym linii.

23 (17H) — **ETB** (ang. end of transmission block), **koniec bloku transmisji.**
Znak służący do wskazania końca bloku danych dla celów transmisji.

24 (18 H) — **CAN** (ang. cancel), **anulowanie.**
Znak wskazujący, że związane z nim dane powinny być pominięte.

25 (19H) — **EM** (ang. end of medium), **koniec nośnika.**
Znak służący do oznaczenia końca nośnika danych.

26 (1AH) — **SUB** (ang. substitute), **zastąpienie.**
Znak używany zamiast znaku uznawanego za nieważny lub błędny.

27 (1BH) — **ESC** (ang. escape), **zmiana zestawu znaków.**
Znak służący do zmiany znaczenia występującego po nim ciągu znaków.

28—31 (1CH—1FH) — **FS, GS, RS, US** (ang. file, group, record, unit separators), **separatory pliku, grupy, rekordu i jednostki.**
Znaki służące do logicznego rozdzielenia informacji zorganizowanej hierarchicznie.

127 (7FH) — **DEL** (ang. delete, rubout), **usuwanie.**
Znak służący do wykluczania innych znaków metodą przebijania.

W porównaniu z nazwami występującymi w normie PN-78/T-42108, zaproponowano niewielkie zmiany. Ponieważ głównym elementem każdej nazwy powinien być w zasadzie rzeczownik w mianowniku, użyłem nazw: **brak informacji, zmiana kodu i powrotna zmiana kodu,** zamiast — bez informacji, poza kodem i w kodzie. Sprecyzowanie znaczenia było powodem wprowadzenia nazw: **wysuw wiersza, odrzucenie i synchronizacja spoczynkowa,** zamiast — zmiana wiersza, zaprzeczenie i synchronizacja. W celu utrzymania zgodności nazw znaków z istniejącym systemem terminów informatycznych, proponuję używanie nazw: **zastąpienie i usuwanie znaku,** zamiast — podstawianie i kasowanie znaku, gdyż wyrazy podstawianie i kasowanie są używane w informatyce w innych znaczeniach niż słowa

dokończenie na str. 17

Węglarz J.: Ocena działania systemów komputerowych — przedmiot, metody, kierunki rozwoju

INFORMATYKA 1987, nr 4, s. 1

Określenie przedmiotu badań oraz omówienie kryteriów, metod i kierunków rozwoju badań w dziedzinie oceny działania systemów komputerowych.

Węglarz J.: Evaluation of computer system performance — subject, methods, development trends

INFORMATYKA 1987, No. 4, p. 1

Qualification of research subject, methods and development trends of computer system performance evaluation.

Węglarz J.: Bewertung von Computersystemleistung — Gegenstand, Methoden, Entwicklungstrends

INFORMATYKA 1987, Nr. 4, S. 1

Darstellung von Forschungsgegenstand, Methoden und Entwicklungstrends der Leistungsbewertung von Computersystemen.

Gurbiel E., Krupicka H., Płoski Z.: Składnia języka Logo — próba formalizacji (1)

INFORMATYKA 1987, nr 4, s. 5

Propozycja formalizacji składni języka Logo na przykładzie dialektu Sinclair-Logo.

Gurbiel E., Krupicka H., Płoski Z.: Logo syntax — a formalization attempt (1)

INFORMATYKA 1987, No. 4, p. 5

Proposal of Logo syntax formalization on example of Sinclair-Logo dialect.

Gurbiel E., Krupicka H., Płoski Z.: Logo-Syntax — ein Formalisierungsversuch (1)

INFORMATYKA 1987, Nr. 4, S. 5

Ein Vorschlag der Logo-Syntax-Formalisierung an Beispiel von Sinclair-Logo-Dialekt.

Kowalski R.: Programowanie logiczne i komputery nowej generacji (2)

INFORMATYKA 1987, nr 4, s. 8

Druga część artykułu, zawierająca przykład zastosowania programowania logicznego w prawodawstwie oraz omówienie innych związków sztucznej inteligencji z inżynierią oprogramowania.

Kowalski R.: Logical programming and new generation computers (2)

INFORMATYKA 1987, No. 4, p. 8

Second part of the paper, which contains example of logical programming application in legislation, as well as discussion of other AI relewance to software engineering.

Kowalski R.: Logische Programmierung und Rechner der neuen Generation (2)

INFORMATYKA 1987, Nr. 4, S. 8

Zweiter Teil des Artikels, der ein Anwendungsbeispiel von logischen Programmierung in Gesetzgebung, sowie Besprechung anderer Zusammenhänge der künstlichen Intelligenz mit Software-Engineering umfasst.

Pawłowski M.: Mikroprogramowanie (1)

INFORMATYKA 1987, nr 4, s. 11

Charakterystyka współczesnych rozwiązań mikroprogramowania układowego.

Pawłowski M.: Microprogramming (1)

INFORMATYKA 1987, No. 4, p. 11

Characteristics of contemporary solutions of microprogram realization.

Pawłowski M.: Mikroprogrammierung (1)

INFORMATYKA 1987, Nr. 4, S. 11

Eine Charakteristik der zeitgenössischen Lösungen von Mikroprogrammierung.

Rażnowiecki A., Syfert A.: Funkcje systemu BIOS dla PC/XT

INFORMATYKA 1987, nr 4, s. 15

Charakterystyka wywołań łączących system operacyjny PC-DOS z systemem wejścia-wyjścia BIOS w komputerach IBM PC/XT.

Rażnowiecki A., Syfert A.: BIOS system functions for the PC/XT

INFORMATYKA 1987, No. 4, p. 15

Characteristics of calls, which connect PC-DOS operating system with BIOS input-output system in the IBM PC/XT computer.

Rażnowiecki A., Syfert A.: BIOS-System-Funktionen für PC/XT

INFORMATYKA 1987, Nr. 4, S. 15

Eine Charakteristik von Aufrufen, die das PC-DOS-Betriebssystem mit dem BIOS-Ein- und Ausgabesystem in IBM PC/XT-Rechner verbinden.

Meller-Kapcia M.: System relacyjnej bazy danych SELKO dla minikomputera MERA 400

INFORMATYKA 1987, nr 4, s. 23

Charakterystyka systemu bazy danych SELKO opracowanego w Instytucie Okrętowym Politechniki Gdańskiej dla minikomputera MERA 400 oraz przykłady zastosowań.

Meller-Kapcia M.: SELKO — a relational data base system for MERA 400 minicomputer

INFORMATYKA 1987, No. 4, p. 23

Characteristics of SELKO data base system, elaborated in the Shipping Institute of the Gdańsk Technical University for MERA 400 minicomputer, and examples of the system application.

Meller-Kapcia M.: SELKO — ein relationales Datenbanksystem für MERA 400-Rechner

INFORMATYKA 1987, Nr. 4, S. 23

Eine Charakteristik von SELKO-Datenbanksystem, das im Institut für Schiffsbau der Technischen Universität in Gdańsk für MERA 400-Kleinrechner erarbeitet wurde, und Beispiele für Systemanwendung.

mercomp

informuje, że świadczy usługi w zakresie:

- rozpoznania potrzeb klienta w zakresie możliwości zastosowań informatyki,
- analizy światowego i krajowego rynku sprzętu komputerowego pod kątem rozpoznanych potrzeb,
- doboru sprzętu komputerowego, określenia konfiguracji,
- wskazania dostawców, analiz cenowych,
- zakupu i kompletacji sprzętu z upoważnienia klienta, odbioru ilościowego i jakościowego dostarczonego sprzętu,
- instalacji i uruchomienia,
- serwisu gwarancyjnego sprzętu,
- serwisu pogwarancyjnego,
- doboru gotowych pakietów oprogramowania,
- tworzenia oprogramowania specjalistycznego,
- modernizacji istniejącego oprogramowania,
- tłumaczenia na język polski dokumentacji technicznych i użytkowych,
- szkolenia użytkowników w zakresie obsługi sprzętu oraz posługiwania się oprogramowaniem,
- innych działań w imieniu i na rzecz zleceniodawcy związanych z informatyzacją jego przedsiębiorstwa.

BLIŻSZYCH INFORMACJI UDZIELA:
Zakład Informatyki CBW MERCOMP
tel. 46-20-51 wew. 388 lub 303

EO/268/87

KONIEC TWOICH PROBLEMÓW

Nareszcie wszystkie potrzebne Ci dane dotrą na czas
w przejrzystej formie tabel i wykresów.

Pakiet oprogramowania na mikrokomputery 16-bitowe

MEGA — BANK

to nowe MEGA możliwości jakie otwierają się przed
Twoim przedsiębiorstwem.

Wyobraź sobie

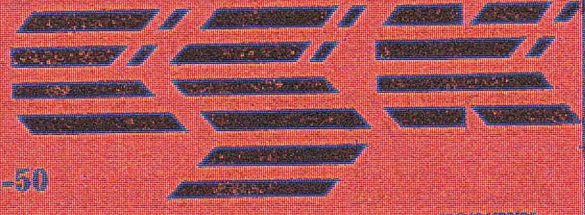
MILIARD

rekordów, które możesz
zapełnić według własnych potrzeb i uznania.

Miliard kooperantów, miliard pracowników, miliard produktów
— z tym wszystkim nasz MEGA-BANK poradzi sobie bez trudu.

System jest łatwy w obsłudze i opracowany w języku polskim.

Gwarantujemy satysfakcję!

COMPUTER STUDIO KAJKOWSCY
PROFESJONALNE OPROGRAMOWANIE MIKROKOMPUTERÓW
ul. Balladyny 3B, 81-524 Gdynia, tel.: 29-00-18, 24-01-50

EO/81/K/86